Paul Levi
Serge Kernbach (Eds.)

COSMOS 7

COGNITIVE SYSTEMS MONOGRAPHS

Symbiotic Multi-Robot Organisms

Reliability, Adaptability, Evolution

Springer

Cognitive Systems Monographs

Volume 7

Paul Levi and Serge Kernbach (Eds.)

Symbiotic Multi-Robot Organisms

Reliability, Adaptability, Evolution

Springer

Rüdiger Dillmann, University of Karlsruhe, Faculty of Informatics, Institute of Anthropomatics, Humanoids and Intelligence Systems Laboratories, Kaiserstr. 12, 76131 Karlsruhe, Germany

Yoshihiko Nakamura, Tokyo University Fac. Engineering, Dept. Mechano-Informatics, 7-3-1 Hongo, Bukyo-ku Tokyo, 113-8656, Japan

Stefan Schaal, University of Southern California, Department Computer Science, Computational Learning & Motor Control Lab., Los Angeles, CA 90089-2905, USA

David Vernon, Khalifa University Department of Computer Engineering, PO Box 573, Sharjah, United Arab Emirates

Editors

Dr. Paul Levi

Professor of Computer Science
IPVS, Universität Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany
levi@ipvs.uni-stuttgart.de

Dr. Serge Kernbach

IPVS, Universität Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany
serge.kernbach@ipvs.uni-stuttgart.de

ISBN 978-3-642-11691-9 e-ISBN 978-3-642-11692-6

DOI 10.1007/978-3-642-11692-6

Cognitive Systems Monographs ISSN 1867-4925

Library of Congress Control Number: 2010925112

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

5 4 3 2 1 0

springer.com

Foreword

It is our pleasure to contribute the foreword to this book on symbiotic multi-robot organisms, which is largely based on the scientific findings and explorations of two major EU research projects, Symbrion and Replicator, funded under the Seventh Framework Programme for Research and Technological development (FP7)[1]. FP7 emphasises consortia of European partners, transnational collaboration, open coordination, flexibility and excellence of research and plays a leading role in multidisciplinary research and cooperative activities in Europe and beyond. Its impact is major in terms of integrating and structuring research communities across national borders to achieve a critical mass, providing the leverage for high-potential fields to take off, and encouraging healthy competition at European level while avoiding unnecessary duplication of research capacities. Research proposals are evaluated through a demanding peer-review process and only the best are selected to be funded by the European Commission (EC). The Information and Communication Technologies (ICT) theme has set out a number of challenges within this context, which cover topics such as cognitive systems, modular robotics, adaptive systems and societies of artefacts.

- **Symbrion** was selected following the Call "Pervasive Adaptation" of the "Future and Emerging Technologies (FET)" programme area[2]. It started on 1 February 2008 and will run for 5 years. FET Proactive addresses evolutionary and revolutionary approaches through multidisciplinary cooperation and investigates new future technology options in response to emerging societal and industrial needs and identifies new drivers for research.
- **Replicator** was selected under the "Cognitive Systems and Robotics" programme area[3]. It started on 1 March 2008 and will also run for 5 years.

[1] The views expressed in this foreword are the sole responsibility of the authors and in no way represent the view of the European Commission and its services.
[2] http://cordis.europa.eu/fp7/ict/fet-proactive/home_en.html
[3] http://www.cognitivesystems.eu

The "Cognitive Systems and Robotics" challenge supports research on the engineering of artificial cognitive systems and in particular endowing robots with cognitive and other advanced capabilities.

These projects address major challenges for the robotics of the future that were identified during intensive consultations with leading scientists. Robots will have to harness and interpret information (e.g., speech, images or sensor data), carry out useful tasks (e.g. manipulation and grasping, exploration and navigation, monitoring and control, situation assessment, communication and interaction) and pursue immediate or long-term goals, autonomously or in cooperation with people or other robots. Moreover future robots will have to be:

- **Reliable and robust**, able to operate in adverse conditions for a long period of time without frequent or major maintenance.
- **Learning**, able to improve their capabilities through individual or social interactions with their environment, people or other robots.
- **Adaptable**, able to operate autonomously in loosely-structured, unpredictable, highly-dynamic and open environments, technological and user contexts.
- **Modular**, able, on their own initiative, to operate as a larger system to tackle problems or carry out tasks they would not be able to tackle or carry out individually.
- **Collaborative and social**, able, on their own initiative, to collaborate in a natural way with people or other robots, and to adapt to each others and to changing needs.
- **Evolvable**, able to elaborate strategies operating at different speeds and time scales, from the short-term development of specific skills to the long-term evolution into different species.

To realise the potential of this exciting field, it is crucial to stimulate contributions from and to create synergies between different disciplines, including robotics, cognitive systems, adaptive systems, the social sciences and biology. Symbrion and Replicator are examples of such efforts. Both projects work towards individual robots that are capable of adapting, reconfiguring and self-assembling into large artificial organisms. They will develop novel principles underlying these robotic organisms, such as self-configuration, self-adjustment and self-learning. Furthermore, they share a common platform to avoid duplication in this area.

- **Replicator** aims at developing small, autonomous reliable, highly capable and sensor-rich (laser, camera, RFID, localisation) robots which are able to self-assemble into large artificial organisms on their own initiative. The bio-inspired evolutionary approach and evolvable hardware structure will enable the robotic organisms to emerge new functionalities, to develop their own cognitive and control structures and, finally, to work autonomously in uncertain situations within open-ended or even hazardous, environments.

- **Symbrion** focuses on evolutionary and bio-mimicking approaches to explore biological concepts with robot populations (e.g., artificial evolution, pervasive evolveability, artificial immunology, genetic self-reprogramming, virtual sexuality). It places more emphasis on the genomic framework and genetic learning based on feedback from the environment. Accordingly, it uses large computation resources on the platform and more emphasis is put on collective behaviours and on the assembly into organisms.

Considering both projects together and their respective funding, 5.4 M€ for Replicator and 5.3 M€ for Symbrion (with a 0.5 M€ extension under negotiation), this is one of the largest combined grants ever in collaborative and evolutionary robotics. In this context the research community, the European Commission and both consortia recognise the crucial importance of identifying key underlying scientific questions, deriving scientific and technological priorities from these goals and using them to obtain important and highly-visible outcomes. Apart from the scientific impact, it is likely that both projects will contribute to longer-term applications in various domains, such as surveillance and intervention, exploration and inspection, and search and rescue.

We hope that the readers will enjoy this book as much as we did, and wish both teams success in achieving their ambitious goals.

Olivier Da Costa
(Replicator Project Officer, DG InfSo "Cognitive Systems, Interaction, Robotics")

Wide Hogenhout
(Symbrion Project Officer, DG InfSo "Future and Emerging Technologies (FET) Proactive")

Acknowledgements

The investigation of artificial multicellularity and the development of robot organisms was only possible by the support of the European Commission, Directorate-General Information Society & Media over several years within the 7th Framework Program in the field of "Cognitive Systems, Interaction, Robotics" and "Future and Emerging Technologies". The Strategic Research Agenda, Vision and Challenges are discussed with many experts from robotics, biology, genetics, evolutionary computations, machine learning. Considerable amount of preparatory works has been done within the I-Swarm, Swarmrobot, GOLEM, NEW TIES, Artificial Culture, LEURRE, SuperBot, Swarm-bots, HYDRA, CEBOT projects as well as discussed within the coordinated actions PerAda (Pervasive Adaptation Research Network) and euCognition (EUCogII – European Network for the Advancement of Artificial Cognitive Systems, Interaction and Robotics), which finally defined the field and the road to implementation. Organizers of the special sections and workshops during CEC, IROS, ICRA, GECCO, ICT, FET and several IEEE conferences gave us a great opportunity to present the ideas to scientific community and to collect a valuable feedback. We are grateful to our multiple reviewers for very intensive and fruitful remarks and corrections as well as to project officers whose continuous support essentially contributed to advances and progress of the research and development.

Ideas, experiments and implementations, described in this book were also impossible without a huge amount of effort and time, invested by young and senior researchers of our consortiums. The work of the last few years was of great strength and we are thankful to the consortiums for readiness to contribute, continuous availability and excellent motivation. This finally allowed formulating the ambitious goals towards possible future ways of the collective robotics.

Contents

List of Contributors

A.E. Eiben
Faculty of Sciences, Department of Computer Science,
VU University Amsterdam,
De Boelelaan 1081a,
1081 HV Amsterdam,
The Netherlands
gusz@few.vu.nl

Amelia Ritahani Ismail
Department of Computer Science, University of York, Heslington, York, YO10 5DD. UK
ritahani@cs.york.ac.uk

Alan F.T. Winfield
Bristol Robotics Laboratory (BRL), University of the West of England, Bristol (UWE), Coldharbour Lane, Frenchay, Bristol BS16 1QY, England
alan.winfield@uwe.ac.uk

Alfons H. Salden
Almende B.V., Westerstraat 50, 3016 DJ Rotterdam, The Netherlands
alfons@almende.com

Andy Tyrrell
Department of Electronics, University of York, Heslington, York, YO10 5DD, UK
amt@ohm.york.ac.uk

Anne C. van Rossum
Almende B.V., Westerstraat 50, 3016 DJ Rotterdam, The Netherlands
anne@almende.com

Arianna Menciassi
Polo Sant'Anna Valdera, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera (PI), Italy
arianna.menciassi@sssup.it

Christoph Hösler
Institute for Evolution and Ecology, University of Tübingen, Auf der Morgenstelle 28, 72076 Tübingen, Germany
hoesler@asoem.org

Christopher Schwarzer
Institute for Evolution and Ecology, University of Tübingen, Auf der Morgenstelle 28, 72076 Tübingen, Germany
christopher.schwarzer@uni-tuebingen.de

Eugen Meister
Institute of Parallel
and Distributed Systems,
University of Stuttgart,
Universitätstr. 38, 70569 Stuttgart,
Germany
Eugen.Meister@
ipvs.uni-stuttgart.de

Evert Haasdijk
Faculty of Sciences, Department of
Computer Science,
VU University Amsterdam,
De Boelelaan 1081a,
1081 HV Amsterdam,
The Netherlands
e.haasdijk@few.vu.nl

Fabio Caparrelli
Materials and Engineering
Research Institute,
Sheffield Hallam University,
City Campus,
Sheffield, S1 1WB,
F.Caparrelli@shu.ac.uk

Florian Schlachter
Institute of Parallel and
Distributed Systems, University of
Stuttgart, Universitätstr. 38, 70569
Stuttgart, Germany
Florian.Schlachter@
ipvs.uni-stuttgart.de

Frances Griffiths
Health Sciences Research Institute,
Warwick Medical School,
University of Warwick, Coventry,
CV4 7AL, UK
F.E.Griffiths@warwick.ac.uk

Guoqiang Fu
Polo Sant'Anna Valdera,
Scuola Superiore Sant'Anna,
Viale Rinaldo Piaggio 34,
56025 Pontedera (PI), Italy
g.fu@sssup.it

Guy Baele
Faculty of Sciences, Department of
Plant Systems Biology,
Bioinformatics and Evolutionary
Genomics Division,
Ghent University, Technologiepark
927, 9052 Ghent, Belgium
guy.baele@psb.vib-ugent.be

Heinz Wörn
Institute for Process Control and
Robotics (IPR), Karlsruhe Institute
of Technology (KIT),
Engler-Bunte-Ring.8,
76131 Karlsruhe, Germany
heinz.woern@kit.edu

Heiko Hamann
Artificial Life Lab of the
Department of Zoology,
University of Graz,
Universitätsplatz 2, 8010 Graz,
Austria
heiko.hamann@uni-graz.at

Hermann Haken
Center of Synergetics,
Institute of theoretical physics,
Pfaffenwaldring 57, IV,
University of Stuttgart,
70550 Stuttgart
cos@itp1.uni-stuttgart.de

Jaouhar Jemai
Ubisense AG, Landshuter Allee 12,
80637 Munich, Germany
jaouhar.jemai@ubisense.net

Jan Dyre Bjerknes
Bristol Robotics Laboratory (BRL),
University of the West of England,
Bristol (UWE), Coldharbour Lane,
Frenchay, Bristol BS16 1QY,

England
jandyre@yahoo.no

Jean-Louis Deneubourg
Center for Nonlinear Phenomena and Complex Systems Campus Plaine - CP 231, Université Libre de Bruxelles B-1050 Brussels - Belgium
jldeneub@ulb.ac.be

Jens Liedke
Institute for Process Control and Robotics (IPR), Karlsruhe Institute of Technology (KIT), Engler-Bunte-Ring.8, 76131 Karlsruhe, Germany
jens.liedke@kit.edu

Jiří Havlík
Institute of Microelectronic Applications, Na Valentince 1003/1, 150 00 Prague 5, Czech Republic
jiri.havlik@ima.cz

Jon Timmis
Department of Electronics and Department of Computer Science, University of York, Heslington, York, YO10 5DD. UK
jtimmis@cs.york.ac.uk

José Halloy
Service d'Ecologie Sociale, Campus Plaine - CP 231, Université Libre de Bruxelles, B-1050 Brussels - Belgium,
jhalloy@ulb.ac.be

Jürgen Stradner
Artificial Life Lab of the Department of Zoology, University of Graz, Universitätsplatz 2, 8010 Graz, Austria
juergen.stradner@uni-graz.at

Kanako Harada
Polo Sant'Anna Valdera, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera (PI), Italy
k.harada@sssup.it

Karel Košnar
Gerstner laboratory, Czech Technical University in Prague (CVUT), Technicka 2, 166 27 Prague 6, Czech Republic
kosnar@labe.felk.cvut.cz

Karl Crailsheim
Department of Zoology, University of Graz, Universitätsplatz 2, 8010 Graz, Austria
karl.crailsheim@uni-graz.at

Leonardo Ricotti
Polo Sant'Anna Valdera, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera (PI), Italy
l.ricotti@sssup.it

Libor Přeučil
Gerstner laboratory, Czech Technical University in Prague (CVUT), Technicka 2, 166 27 Prague 6, Czech Republic
preucil@labe.felk.cvut.cz

Lutz Winkler
Institute for Process Control and Robotics (IPR), Karlsruhe Institute of Technology (KIT), Engler-Bunte-Ring.8, 76131 Karlsruhe, Germany
lutz.winkler@kit.edu

Maizura Mokhtar
Department of Electronics, University of York, Heslington,

York, YO10 5DD. UK
mm520@ohm.york.ac.uk

Marc Szymanski
Institute for Process
Control and Robotics (IPR),
Karlsruhe Institute of
Technology (KIT),
Engler-Bunte-Ring.8,
76131 Karlsruhe, Germany
marc.szymanski@kit.edu

Marc Schoenauer
TAO project-team, INRIA
Saclay – Île-de-France and
CNRS UMR 8623,
LRI, Université Paris-Sud,
91405 Orsay Cedex, France
Marc.Schoenauer@inria.fr

Martin Velek
Institute of Microelectronic
Applications, Na Valentince 1003/1,
150 00 Prague 5, Czech Republic
martin.velek@ima.cz

Michèle Sebag
TAO project-team, CNRS UMR 8623
and INRIA Saclay – Île-de-France,
LRI, Université Paris-Sud,
91405 Orsay Cedex, France
Michele.Sebag@lri.fr

Nicolas Bredeche
TAO project-team,
INRIA Saclay – Île-de-France
and CNRS UMR 8623,
LRI, Université Paris-Sud,
91405 Orsay Cedex, France
Nicolas.Bredeche@lri.fr

Nick Owens
Department of Electronics,
University of York,
Heslington, York,
YO10 5DD. UK
ndlo100@ohm.york.ac.uk

Nico K. Michiels
Institute for Evolution and Ecology,
University of Tübingen,
Auf der Morgenstelle 28,
72076 Tübingen,
Germany
nico.michiels@uni-tuebingen.de

Olga Kernbach
Institute of Parallel and
Distributed Systems,
University of Stuttgart,
Universitätstr. 38, 70569 Stuttgart,
Germany
Olga.Kernbach@
ipvs.uni-stuttgart.de

Oliver Scholz
Fraunhofer Institute for
Biomedical Engineering,
Ensheimer Str. 48,
66386 St. Ingbert, Germany
Oliver.Scholz@ibmt.fraunhofer.de

Paolo Dario
Polo Sant'Anna Valdera,
Scuola Superiore Sant'Anna,
Viale Rinaldo Piaggio 34,
56025 Pontedera (PI), Italy
paolo.dario@sssup.it

Paolo Corradi
Polo Sant'Anna Valdera,
Scuola Superiore Sant'Anna,
Viale Rinaldo Piaggio 34,
56025 Pontedera (PI), Italy
paolo.corradi@sssup.it

Paul Levi
Institute of Parallel and
Distributed Systems,
University of Stuttgart,
Universitätstr. 38,
70569 Stuttgart, Germany
Paul.Levi@ipvs.uni-stuttgart.de

Petr Štěpán
Gerstner laboratory,

Czech Technical University
in Prague (CVUT),
Technicka 2,
166 27 Prague 6,
Czech Republic
stepan@labe.felk.cvut.cz

Raja Humza Qadir
Fraunhofer Institute for
Biomedical Engineering,
Ensheimer Str. 48,
66386 St. Ingbert, Germany
rajahamza@gmail.com

Ran Bi
Department of Electronics,
University of York,
Heslington, York,
YO10 5DD. UK
rb507@ohm.york.ac.uk

Rene Matthias
Institute for Process Control
and Robotics (IPR),
Karlsruhe Institute of
Technology (KIT),
Engler-Bunte-Ring.8,
76131 Karlsruhe, Germany
rene.matthias@kit.edu

Ronald Thenius
Artificial Life Lab of the
Department of Zoology,
University of Graz,
Universitätsplatz 2,
8010 Graz, Austria
ronald.thenius@uni-graz.at

Salah Karout
Materials and Engineering Research
Institute, Sheffield Hallam
University, City Campus,
Sheffield, S1 1WB
S.Karout@shu.ac.uk

Sergej Popesku
Institute of Parallel and
Distributed Systems,
University of Stuttgart,
Universitätstr. 38,
70569 Stuttgart,
Germany
Sergej.Popesku@
ipvs.uni-stuttgart.de

Serge Kernbach
Institute of Parallel and
Distributed Systems,
University of Stuttgart,
Universitätstr. 38,
70569 Stuttgart, Germany
Serge.Kernbach@
ipvs.uni-stuttgart.de

Stephen McKibbin
Materials and Engineering
Research Institute,
Sheffield Hallam University,
City Campus, Sheffield,
S1 1WB
S.McKibbin@shu.ac.uk

Ted P. Schmidt
Almende B.V.,
Westerstraat 50,
3016 DJ Rotterdam,
The Netherlands
ted@almende.com

Thomas Schmickl
Artificial Life Lab of the
Department of Zoology,
University of Graz,
Universitätsplatz 2,
8010 Graz, Austria
thomas.schmickl@uni-graz.at

Tomáš Krajník
Gerstner laboratory,
Czech Technical University
in Prague (CVUT), Technicka 2,
166 27 Prague 6, Czech Republic
tkrajnik@labe.felk.cvut.cz

Vladimir Red'ko
Center of Optical Neural Technologies of Scientific Research Institute for System Analysis, Russian Academy of Sciences, Vavilova St., 44/2, 119333 Moscow, Russia
vgredko@gmail.com

Wenguo Liu
Bristol Robotics Laboratory (BRL), University of the West of England, Bristol (UWE), Coldharbour Lane, Frenchay, Bristol BS16 1QY, England
wenguo.liu@brl.ac.uk

Yao Yao
Faculty of Sciences, Department of Plant Systems Biology, Bioinformatics and Evolutionary Genomics Division, Ghent University, Technologiepark 927, 9052 Ghent, Belgium,
yao.yao@psb.vib-ugent.be

Yves Van de Peer
Faculty of Sciences, Department of Plant Systems Biology, Bioinformatics and Evolutionary Genomics Division, Ghent University, Technologiepark 927, 9052 Ghent, Belgium,
yves.vandepeer@psb.vib-ugent.be

Acronyms

ACD	Adaptive Critic Design
AI	Artificial Intelligence
AIS	Artificial Immune System
ALS	Active Localization System
ANN	Artificial Neural Network
AHHS	Artificial Homeostatic Hormone System
ART	Adaptive Resonance Theory
CAL3D	Character Animation Library 3D
CAN	Controller-Area Network
CAPI	Communication Application Interface
COP	Constraint-Optimization Problem
CSP	Constraint-Satisfaction Problem
CS	Chip Select
CNT	Carbon Nanotube
DE	Difference Equations
DOF	Degrees Of Freedom
DPM	Dynamic Power Management
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EM	Energy Management
EMa	Evolution Map
EMS	Energy Management System
ES	Evolution Strategy
ESN	Echo-State Network
FMEA	Failure Modes and Effects Analysis
FSM	Finite State Machine
GA	Genetic Algorithm
GPU	Graphics Processor Unit
GRN	Gene Regulatory Network
GWT	Global Workspace Theory
HAL	Hardware Abstraction Layer

I2C	Inter-Integrated Circuit
IDA	Intelligent Distribution Agent
IKN	Itti-Koch-Niebur
IR	Infra-RED
LM	Lagrange Multiplier
LSM	Liquid State Machine
MAS	Multi-Agent Systems
MCP	Chemo-taxis Protein
MFC	Microbial Fuel Cells
MRC	Master Robot Cell
ML	Machine Learning
MLP	Multilayer Perceptron
MPU	Micro Processing Unit
MTBF	Mean time between failures
ODE	Ordinary Differential Equations
ODE	Open Dynamic Engine
OSG	Open Scene Graph
PID	ProportionalIntegralDerivative e.g. controller
PFSM	Probabilistic Finite State Machine
PCB	Printed Circuit Board
RFID	Radio Frequency Identification
RTOS	Real Time Operating System
RTI	Runtime-Infrastructure
QAP	Quadratic Assignment Problem
QFT	Quantum Field Theory
SO	Self-Organization
SOM	Self-Organized Maps
SMPA	Sense-Model-Plan-Act
SPI	Serial Peripheral Interface
TCR	T Cell Receptor
TOF	Time-of-Flight
UWB	Ultra-Wideband
WM	World Model

Introduction

The main question of this book is related to one of the greatest moments in the history of life – appearance of multicellular structures. The rise of multicellular from unicellular is a huge evolutionary step, however we do not exactly know how multicellular organisms appear and which mechanisms take part in this phenomenon. We know multicellular organisms are self-adaptive, self-regulative and self-developing, however we do not know its evolutionary origin and developmental organization.

Robotics is seen by many researchers only as a tool to improve productivity. However, robotics can also be used as an instrument to explore unknown and unclear issues in nature, to model elements of organic life, to experiment with self-development and evolution, and to propose plausible explanations. The great vision, which consolidates many interdisciplinary researchers, is a vision of self-adaptive, self-regulative and self-developing robots that reflect multicellularity in nature – a vision of artificial robot organisms. Like multicellular beings, these artificial organisms consist of many small cell-modules, which can act as one structure and can exchange information and energy within this structure. Moreover, these structures can repair themselves and undergo evolutionary development from simple to complex organisms.

The vision of artificial multicellularity is very challenging. For design and production of mechatronic artificial cells, experts in robotics are required. An artificial organism, seen technically, is a large distributed computational system – it requires software engineers to program a basic functionality as well as to develop computer simulations. Organisms can autonomously harvest environmental energy and distribute it between cells – this is a deal for specialists in energetic homeostasis. Each of these cell-modules possesses dozens of different sensors. When many modules get integrated into one system, the problems of sensor fusion, distributed sensor processing, world modeling and developing a cognitive functionality become a reality. Adaptation requires aspects in machine learning and in adaptive control. Following a genetic bio-inspiration, artificial organisms possess a genome – this involves biologists, specialized in genetics. Artificial organisms should protect themselves – it requires an artificial immune system. To handle genetic issues, there is a demand on

specialists in artificial evolution – computer scientists, as well as in natural evolution – ecologists and reproduction biologists. Finally, collectively working robots create social structures and are capable for evolving pre-semantic languages – this requires researchers of artificial cultures. Thus, the vision of artificial organisms created one of the largest interdisciplinary consortiums, funded by European Commission, to investigate the way of artificial multicellularity.

Technological exploitation of multicellularity provides different practical advantages not only for advanced robotics, but also for autonomous and adaptive systems in general. Three most important advantages are extended reliability, advanced adaptivity and self-evolving properties. They are essential technological and conceptual breakthroughs and are reflected in the title of this book "Symbiotic Multi-Robot Organisms: Reliability, Adaptivity, Evolution".

Reliability in general context is related to the ability of a system to work durably in different hostile or unexpected circumstances. Normally, a high reliability is achieved in two ways, by making the systems very robust, as well as by introducing backup systems, working in parallel. Multicellular organisms in nature utilize another principle of reliability: when some cells are destroyed, they are replaced by other cells, which overtake their functionality. When applying this concept to an artificial organism, a part of (or the whole) organism should first self-disassemble, the destroyed cell-modules should be removed, and then an organism self-assembles again. Capabilities of basic robot modules for autonomous self-assembling and for dynamic change of functionality are key points of the extended reliability.

Adaptivity is another key feature of advanced autonomous systems and indicates an ability of a system to fit to a changing environment. There are different adaptive mechanisms, as well as different degrees of adaptivity. Multicellularity introduces a new component into adaptive processes – morphogenesis – the self-development of structure, functionality and behavior during a life cycle of the organism. By changing the structure, an organism can dynamically create such a functionality and behavior, which are best suited to the given condition of an environment. Morphogenesis and morphodynamics represent a new understanding of adaptivity in complex systems.

Both reliability and adaptivity mean a high developmental plasticity, where an organism can dynamically change itself, modify its own structural and regulatory components. As observed in nature, the developmental plasticity is a necessary condition for evolutionary processes – such processes, which can potentially make a system more complex, increase information capacity and processing power. Artificial organisms also possess a relatively high degree of developmental plasticity (however much lower than biological organisms) and can demonstrate to some extent an artificial evolution – analogous to natural evolution – process which can potentially make organisms more intelligent. Such an artificial evolution in real robotic systems is a great challenge not only in technological aspects, but also in our understanding of a long term-controllability of self-developmental systems.

Not only does an artificial evolution represent a challenge for researchers and engineers, but also other theoretical and technological aspects are very demanding or generally unclear. The first essential challenge is a good engineering of mechatronic

cell-modules, which should demonstrate 2D locomotion on a surface, 3D actuation within a heavy organism, autonomous docking to each other, large on-board energy resources, different sensors and sufficient computation/communication. Of utmost importance is that the modules should be small in size and light in weight.

Cognitive functionality of artificial organisms is another essential challenge. Classical approaches of technical cognition should be supplemented by bio-inspired mechanisms, where the sensor information is directly integrated into high-level regulative structures. Moreover, cognitive functionality should take into account a morphogenetic embodiment, especially when a capability to sense and to act is changing along a self-concept. The regulative mechanisms include artificial hormone, immune and neural systems as well as different approaches from the theory of optimal control. Here we encounter several open problems such as an optimal combination of bio-inspired and tech-inspired mechanisms, self-programming capabilities of high-level controllers or conflicts between adaptive and regulatory functionality.

Finally, returning to artificial evolution, there are several conceptual challenges related to dependencies between evolution and learning, between evolution and self-organization, appearance of common genome in the organism from a genetic cloud in swarm, sexual and asexual reproductive ways and other points. Due to the developmental plasticity, artificial organisms are potentially capable of a long-term unbound, open-ended evolution. Collective systems, evolved in groups, demonstrate social behavior, initial forms of pre-semantic languages and are so biased from initial design goals. Such a long-term self-developmental process is a great challenge for researchers in terms of its controllability and predictability and has serious consequences for future coexistence of natural and artificial systems.

Structure of the book

The structure of this book follows the challenges and has five thematic chapters. The *first chapter* is devoted to a common theoretical background of artificial organisms. It overviews swarm and organism modes, functional and structural self-organizations taking place during morphogenesis, as well as introduces adaptive and self-developmental processes in Sect. 1.1. Sect. 1.2 deals with general information properties of artificial self-developmental systems, it utilizes quantum field theory as an instrument for measuring sematic information and developing interactions and dynamics of multicellularity. Sect. 1.3 treats the swarm mode of the organism and introduces several approaches for mathematical modelling of collective robotic systems. Based on these models, failure, reliability and scalability properties are analyzed.

The *second chapter* introduces several hardware issues which are relevant for further consideration. Heterogeneous mechanical platforms are described in Sect. 2.1. It presents three different basic modules, composing artificial organisms, active and passive tools as well as principles of docking. Sensing, computation and communication are described in Sect. 2.2. This section also provides an overview of a common architecture. Issues of energetic autonomy and energy harvesting are examined

in Sect. 2.3. Attention is paid not only to energy management and distribution in the organism, but also to different means of harvesting energy from the environment. The final Sect. 2.4 is devoted to hardware and 3D-software simulation and addresses issues such as a representation of kinematic chains, using L-systems for dynamic modelling and predicting morphogenetic capabilities of artificial organism in simulation.

The *third chapter* covers cognitive aspects of artificial organisms from different perspectives. Sect. 3.1 is devoted to world modelling and cognition from the viewpoint of classical AI. Complementary to this section, several ideas of a new AI approach (e.g. embodied cognition) are introduced in Sect. 3.2. This section studies also sensor-fusion and sensomotor awareness. Finally, Sect. 3.3 adds more general points of self-organization, probabilistic and multi-agent systems into artificial cognition.

The *fourth chapter* is one of the largest in the book and introduces dedicated approaches for controlling artificial organisms. This chapter has bio-inspired and tech-inspired parts. An overview of different bio-inspired controllers is given in Sect. 4.1. Hormone-based control and hormonal homeostasis are considered in Sect. 4.2. Embryological aspects, morphogenesis, controller chapping and cell-specialization are introduced in Sect. 4.3. Finally, artificial immune systems are considered in Sect. 4.4. The tech-inspired part of fourth chapter starts in Sect. 4.5, which consider morphogenesis as constraint assignment problem and proposes several self-organizing mechanisms for self-assembling. Adaptive kinematics based in the crew theory are introduced in Sect. 4.6, which can be applied for on-line and on-board control of collective locomotion.

The *fifth chapter* considers learning and artificial evolution in swarm and in organism modes. It begins with machine learning in Sect. 5.1. Social learning and computational evolutionary approaches are reviewed in Sect. 5.2. In this section, attention is also paid to genetic issues such as a structure of genome and evolving of controllers. Sexual mechanisms for artificial evolution and more general issues of artificial sexuality are considered in Sect. 5.3. Asexual virus-like behavior in the genetic cloud of swarms is introduced in Sect. 5.4, where several self-learning approaches are considered. Social behavior and memetic evolution are finally concluded in Sect. 5.5, which introduces artificial cultures emerging in groups of autonomous self-developing robots.

Stuttgart, December 2009

Paul Levi,
Serge Kernbach

Chapter 1
Concepts of Symbiotic Robot Organisms

1.1 From Robot Swarm to Artificial Organisms: Self-organization of Structures, Adaptivity and Self-development

Serge Kernbach

Collective systems possess very interesting properties. They are flexible, reliable, have extended capabilities for adaptation, self-organization and self-development. Many natural systems, such as atomic or molecular phenomena (Balzani *et al.*, 2003), social insects or animals (Camazine *et al.*, 2003) are collective on the level of their aggregation or population. Since these survived millions of years in the course of multiple evolutionary processes, we can learn from them how to achieve long-term stability, diverse functionality and reliability for artificial collective systems.

In technical progress, in particular in robotics (Siciliano & Khatib, 2008), artificial collective systems are also a focus of research and technological development. However, due to a specific structural and functional organization, collective systems represent several essential challenges for researchers. We can emphasize three most important challenges whose solutions may contribute not only to technological advancements but also to theoretical understanding of underlying processes in collective systems. These challenges are structural adaptability, evolvability and self-development and, finally, a long-term independency of these systems.

Structures in natural systems, for example protein structures of bio-molecular systems (Alberts *et al.*, 2008) or social structures in groups of animals, are a major factor in defining a collective functionality and finally a collective behavior. Features of individuals are still important, however a collective is capable of demonstrating a functionality not available to individuals. Designers of artificial systems are able to create not only functions and behavior, but also to define new structures (Kernbach, 2008). The interplay between structures, functions and behavior allows multiple self-organizing and self-developmental processes. Many holistic and reduction approaches have been suggested to deal with this problem, such as classical and modern control theory (Kolesnikov, 1994), methods

P. Levi and S. Kernbach (Eds.): Symbiotic Multi-Robot Organisms, COSMOS 7, pp. 5–77.
springerlink.com

from distributed AI and multi-agent systems (Weiss, 1999), bio-inspired solutions (Floreano & Mattiussi, 2008) and synergetics (Haken, 1988). However, due to an emergence on functional and behavioral levels, the complexity of such structural systems is very high. This, and a lack of understanding of structural phenomena, often hinders researchers in realizing desired properties of artificial systems by using structure-function dependencies.

Adaptive and self-developmental processes in collective systems happen on different levels and are defined in a wide range from adaptation, self-defence and self-healing to unbounded self-evolving. These are very attractive and desired properties allowing systems to develop themselves from simple to complex ones, to increase their own functional diversity and improve their own control. Working with these processes in artificial systems, we encounter several problems. First, technical collective systems are driven by two different forces: design goal and adaptive fitness. Such issues as long-term controllability, predictability and validation are the focus of research here. Secondly, biological concepts of adaptation and self-development are valid for populations, involving such processes as death, birth, reproduction and others, which are not very natural for robotics. Therefore we are looking for such approaches, which can be applied even for a single robot, do not require very powerful computational resources and may be utilized in short-term operational situations. We call them *on-line* and *on-board* self-developmental processes. It is obvious, that an exploration of new techniques, for example a combination of evolution and self-organization (so called "evolutionary self-organization") or traditional AI-based decision making, planning or learning with bio-inspired approaches, is required.

Long-term independency and is an integrating property of many other factors. It can be understood as a capability to work independently a long time period without the need for continuous human maintenance and supervision. This property depends largely on reliability and good engineering of the system, on capabilities of regulatory autonomy to deal with unbounded issues in self-development, and can be considered as the most challenging task in autonomous robotics.

Research in collective robotics is intensively addressing these problems. One methodological approach is represented by networked robotics (Kumar *et al.*, 2008). Networked robotics assumes that essential communication resources are necessary for problem solving. Another alternative is given by swarm robotics (Parker, 2008), which involves interactions instead of communication. Both approaches utilize individual functionality of robots in creating *behavioral emergence*, i.e. *individual functionality* $\rightarrow$ *emergent collective behavior*. This scheme can address all three challenges from the viewpoint of functions and behavior, such as exploring social hierarchies, ecological dynamics or population-based evolution. However, an essential problem still remains untouched by these systems; they cannot build structures, i.e. no structural level is available.

The structural level is important for investigating the generating dependencies such as *collective structures* $\rightarrow$ *emergent collective functionality* $\rightarrow$ *emergent collective behavior*. This reflects the very important role of *functional emergence* which appears in such processes as embryogenetic, morphogenetic and ontogenetic development, cell differentiation, intrinsic evolution in robot systems and others.

To investigate dependencies between structures and functions, we need a new class of robot systems having a cellular-like structure and capable of autonomous self-assembling into organisms. Since these new systems possess self-developmental features, it is expected that several system-relevant (i.e. not relevant for individual robots) mechanisms and functions will emerge. A capability of artificial organisms to modify their own morphology and size means that such mechanisms and functions should be very flexible, scalable and be implemented in a specific cooperative way, i.e. without essential centralization. Such cooperation between different regulative, structural and behavioral aspects is the central issue of artificial organisms. For a similar co-dependent functionality of natural systems, the notion of symbiosis is used. To emphasize the cooperative aspect of structural self-developmental phenomena in these systems, we call them *symbiotic multi-robot organisms*.

Symbiotic multi-robot organisms allow us to address the challenges of collective robotics from the viewpoint of structures and developmental plasticity. This increases the number of available degrees of freedom for emerging control, where the system can change its own structures to adapt to its environment. This enables new structural self-organization and can involve bounded and unbounded self-developmental processes. In general, this can result in extended reliability, adaptability and long-term independence of such artificial systems. In addition to new technology, this may lead to deeper understanding the phenomena of collective intelligence and artificial evolution.

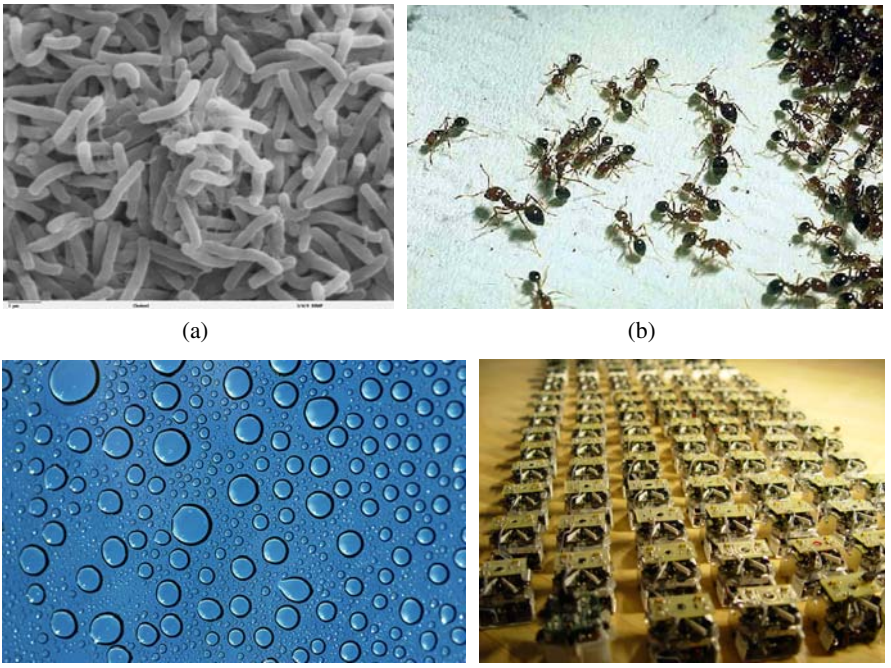
1.1.1 Mono- and Multi- functional Artificial Self-organization

Collective systems consist of many independent interacting elements, we can find them in living and nonliving nature, see Fig. 1.1(b,c); at all scales: from nano- and micro-scales, such as bacteria in Fig. 1.1(a) to large-scales such as galaxies. Currently, we stet a growth of different artificial collective systems, see Fig. 1.1(d).

Collective systems possess many amazing properties and phenomena, which fascinate researchers. These systems scale well and are very reliable, they possess different self-regulating mechanisms, they are capable of self-organization and emergent phenomena – when ordered macroscopic behavior emerges from interactions among microscopic elements. Macroscopic behavior is often visible as different dynamic or static patterns, as shown in Fig. 1.2. When representing the structure of collective systems, we have to draw two representation levels:

microscopic level of consideration is the level of interacting elements (Fig. 1.1), where individual behavior is the focus and collective properties of the system are not observable; **macroscopic level of consideration** is the level where the whole collective behavior (Fig. 1.2) is visible to an external observer and properties of individuals are neglected.

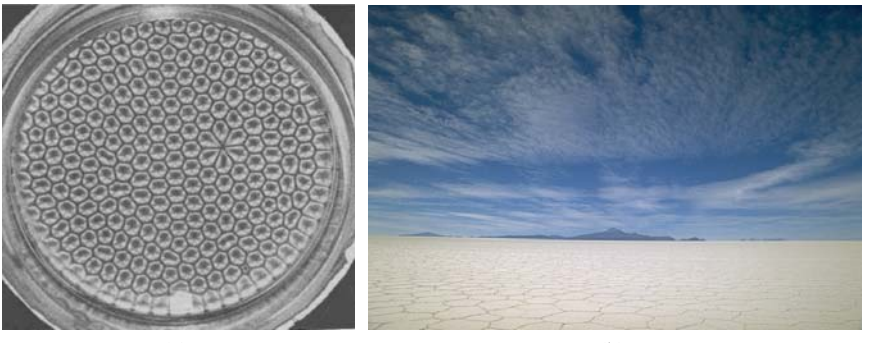
One of the key problems in collective systems is that *"we cannot proceed directly from microscopic to macroscopic level, i.e. from individual models to emergent*



(a) (b)

(c) (d)

Fig. 1.1 Examples of collective systems – microscopic level of consideration; **(a)** Vibrio cholera bacteria in SEM micrograph (courtesy of Patrick Hunt and Andreea Seicean phunt@stanford.edu); **(b)** Swarm of ants; **(c)** Water droplets on glass; **(d)** Artificial swarm of micro-robots Jasmine.



(a) (b)

Fig. 1.2 Macroscopic patterns emerge as a result of different self-organizing processes – macroscopic level of consideration; **(a)** Benard-cells, (from presentation of E. Laurien, Rayleigh-Benard-Konvektion, University of Stuttgart,Germany); **(b)** Patterns in nature (with permission of Bernhard Mühr).

collective behavior. If we need some specific collective behavior, we do not know which individual models can produce it" (Arnold, 1988, p.212). The main reason is the enormous complexity generated by interactions among components. Each interaction step creates a new correlation cascade and this dramatically increases the total complexity (Prigogine, 1962). However, complexity in collective systems is distributed in different way. To describe it, the notion of operational principle is introduced (Kernbach, 2008).

Collective systems with **vertical operational principle** have strong hierarchies in their organization: elements on the lowest-level are ruled by a few elements on higher levels. The organization and distribution of complexity looks like a pyramid: high complexity below and a low complexity on the top. Collective systems with **horizontal operational principle** do not have hierarchical organization, their complexity is similar on all levels of abstraction.

Operational principles are directly related to the problem of structures and control (Turchin, 1977). Collective systems with the vertical operational principle utilize hierarchical control (Vinter, 2000). Systems with horizontal operational principle use self-organizing control mechanisms (usually away from instabilities) without growing hierarchies. This makes the development of such mechanisms more difficult, however horizontal systems are very scalable and reliable.

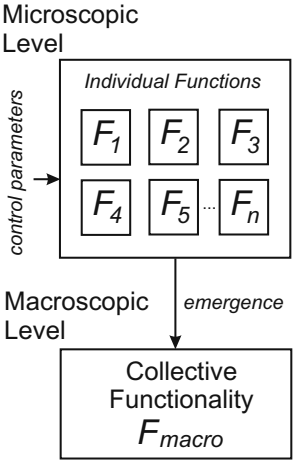


Fig. 1.3 Two levels of consideration – relation between functions and behavior.

Elements on the microscopic level behave according some local interaction rules, we denote them as functions F_i, see Fig. 1.3. Moreover, all of F_i are on the same level of hierarchy, in other words there is no explicit control in this system. F_i have some control parameters, which can depend on environmental conditions as well as on internal parameters. On the macroscopic level we observe a collective functionality F_{macro}. There are two types of F_{macro}. Let us consider the Fig. 1.4(a). This is a typical heterogeneous collective traffic system, performing a rescue operation on the highway. Combining microscopic functionality F_i of each vehicle in this system, we can observe a multitude of F_{macro}, such as collective transportation, different rescue missions and others. In Fig. 1.4(b) we show another collective system – artificial salamander, developed in biologically Inspired Robotics Group, EPFL (Chevallier *et al.*, 2008). Each segment of this system is autonomous in terms of behavioral activities and is synchronized with other segments

(a) (b)

Fig. 1.4 Examples of collective systems with mono- and multi- functional F_{macro}; **(a)** Heterogeneous multi-functional collective system; **(b)** Artificial salamander (biologically inspired robotics group, EPFL), demonstrating mono-functional behavior (Chevallier *et al.*, 2008).

through bio-inspired signal transmission. All segments are connected to each other and the whole system can demonstrate a functional pattern of specific anguine-like locomotion.

F_{macro} is **multi-functional**, when it can demonstrate many different functional patterns. Moreover, diversity degree of F_{macro} depends on heterogeneity and a common number of elements. F_{macro} is **mono-functional**, when it can demonstrate only one homogeneous functional pattern, parameterized by one or several parameters.

Both, mono- and multi- functional systems possess several advantages and disadvantages. Most mono-functional systems (both collective and not collective) are cheaper and simpler from a control perspective. When we consider modern manufacturing such as flexible (Qiao *et al.*, 2006) or reconfigurable manufacturing systems (Galan *et al.*, 2007), we encounter mostly only mono-functional systems. In general, mono-functional systems are more attractive for creating technically useful behavior. Disadvantages of mono-functional systems are low reliability and scalability compared to multi-functional systems.

Collective functionality F_{macro} can be random, chaotic and, in several cases, can represent an ordered (or synchronized) pattern. This ordered collective behavior may have forms of symmetrical patterns, as shown in Fig. 1.2 or, in general, it may be synchronized in spatial, temporal and functional ways. The process, leading to ordered macroscopic behavior F_{macro} through interactions between F_i, is denoted as self-organization. "*The self-organization is a process by which global external influence stimulate the start of internal for the system mechanisms, which bring forth the origin of specific structures in it*" (Bushev, 1994, p. 24). We will denote this process as **functional self-organization**. In artificial collective systems, the designer can change interaction among elements and thereby modify their collective properties.

All F_i can be created in such a specific way, that collective behavior has an ordered character – this can be denoted as **artificial functional self-organization**. As already mentioned, the problem of creating purposeful self-organization, as well as a general problem of emergent phenomena, is one of complexity. There is no way to predict analytically such rules F, which will lead to the desired collective behavior. However, self-organization possesses several advantages making this phenomenon attractive in practical applications:

- *Flexibility of self-organized collective behavior.* Collective behavior in artificial systems can be easily changed by modifying interactions. Even a small change, e.g. around critical points, can qualitatively change the whole collective behavior. The mechanisms of "adaptive self-organization" can provide a high degree of flexibility.

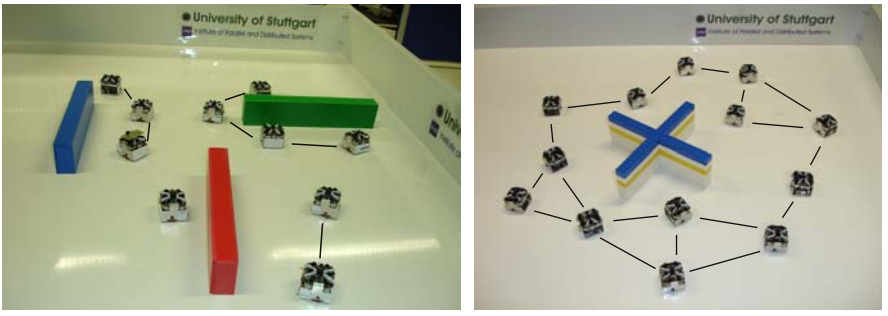
- *Reliability and Scalability.* Since all F_i are on the same level of hierarchy, some elements can be removed (destroyed) without changing collective functionality. Scalability of collective systems is based on the same principle. When self-organized mechanisms provide scalability for load parameters, like number of elements or diversity (Constantinescu *et al.*, 2004), collective systems may be scalable or even super-scalable, see Sect. 1.3.

In the following section we extend the notion of self-organization for the structural case.

1.1.2 Collective Robotics: Problem of Structures

Consider now self-organization in technical collective systems, in particular in collective robotics, we remark that these systems possess additional degree of freedom. They are able to change their own macroscopic structures. These structures are topologies of information networks, neighborhood connectivity or even 3D structures. We consider an example of such structures based on collective perception in a robot swarm, as shown in Fig. 1.5. To recognize large objects, small swarm robots should create a network around the object of interest. Important are not only topologies (open, closed, chain-like, star-like), but also connectivity of robots because it results in different object recognition algorithms running in each robot (Pradier, 2005). In Fig. 1.5 we can observe two different networks, open-chain in Fig. 1.5(a) in experiments with color perception in swarm (Zetterström, 2006) and surrounding the object in Fig. 1.5(b) in collective perception of large objects (Kornienko *et al.*, 2005). In this way, robots have several structural rules of how to create different networks. In turn, the emerging structure influences the local functional rules in each robot and, finally, the whole swarm demonstrates different collective perception behavior.

Normally, structures and functionality are closely related to each other. By changing macroscopic structure, the system also changes its own functionality and correspondingly behavior (Kernbach, 2008). The relation between structures, functions and behavior can be represented as shown in Fig. 1.6. We denote this relationship as "generating" because the upper level generates the lower level, i.e. structures generate functions and functions generate behavior. However, the relation between structures and functions is non-trivial and several types of this relation are observed.



(a) (b)

Fig. 1.5 Examples of different network structures in collective perception, lines show communication between robots; **(a)** Experiments in color perception in swarm, robots create open-chain network and recognize an object by a feature matching (Zetterström, 2006); **(b)** Collective perception of large objects, robots create close-chain network and recognize an object by a probabilistic approach (Kornienko *et al.*, 2005).

In the first case, such as collective perception in Fig. 1.5, structural rules generate functional rules. For example, the structure of the network is defined by structural rules, by the number of locally achievable robots and by geometry/size of the object. In turn, the amount of information flowing from robot to robot depends on the topology. Each robot adapts its own processing rules (i.e. functionality) to this information flow. Collective perception is finally defined by combination of different processing rules.

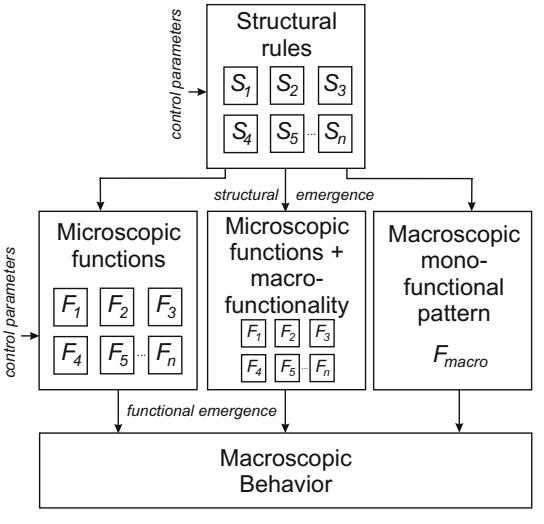
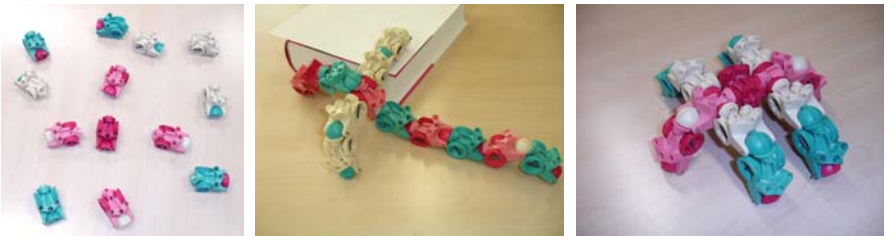


Fig. 1.6 Relation between structures, functions and behavior.

In this way, collective behavior is defined by interacting individual functionalities, in turn, defined by the structure of the network. More generally, we can observe here two emergent processes at functional and structural levels with multi-functional collective activity.

In the second case, elements create different structures with mono-functional macroscopic functionality. This functionality ultimately demonstrates a behavior. A



(a) (b) (c)

Fig. 1.7 Different structures and functionality; **(a)** Initial not aggregated modules; **(b)** Aggregated snake-legged structure and resulting crawling locomotive functionality; **(b)** Aggregated 6-leg structure and resulting legged functionality.

famous example are L-systems, proposed by Aristid Lindenmayer in 1968 (see e.g., (Prusinkiewicz & Hanan, 1980)). The central notion of L-systems is the concept of rewriting, i.e. successive replacing a simple initial object using a set of rewriting rules. These replacing rules can be viewed as structural rules, producing different tree-like fixed patterns (they can be viewed as a mono-functional activity), see more in Sect. 2.4. Robots, which are able to dock to each other and create 3D functional organisms, are another example of mono-functional macroscopic functionality. Topological models of such robots are shown in Fig. 1.7(a). These models have rotational and bending degrees of freedom (Kornienko *et al.*, 2007). Following specific connection (structural) rules, they can create different macroscopic structures, such as snake-like or 6-legged systems, see Fig. 1.7(b-c). As we observe, these 3D structures possess only mono-functional locomotion, defined by the corresponding snake-like or legged motion principle.

Considering Fig. 1.7(b-c), we should take into account not only spatial functionality, but also diverse sensing, homeostatic, energetic and other processes, which will run in real robots. Depending on a spatial position, modules can specialize in performing different tasks. For example, in the topological model of a 6-leg organism shown in Fig. 1.7(c), we can imagine that only a few elements will perform actuation (e.g. they specialize as active joints), elements in the middle take a role of information processing, there are sole-, front- or back- sensor elements. We observe in this case a combination of emergent and macro-functional approaches, as shown in Fig. 1.6.

By analogy with functional self-organization, we define **structural self - organization** as a process leading to emergence of different microscopic and macroscopic functional patterns, which, in turn, emerge as collective phenomena on behavioral level. Since in artificial systems corresponding structural and functional rules can be changed, we denote self-organization in such systems as **artificial structural self-organization**. In Table 1.1 we collect several characteristics of collective systems capable of structural phenomena. The main difference between functionally and structurally self-organizing systems consists of a higher developmental

Table 1.1 Several characteristics of collective systems capable of structural phenomena.

Level	Advantages	Problems
Regulatory	Self-regulation	Long-term stability
	Internal homeostasis and self-healing	Possible communication overhead
	High developmental plasticity	Long-term controllability
Structural	High reliability and scalability	
	Dynamical change of structures	Predictability of functional emergence
Functional	Mono-functional behavior	
	Dynamical change of functionality	Predictability of behavioral emergence
	Emergence of functionality	Difficulties with analytical prediction
Behavioral	New class of adaptive behavior	Double emergence

plasticity in the last case. In the next section we consider using this plasticity for adaptation and self-development.

1.1.3 Adaptability and Self-development

In previous sections we considered collective systems capable of structural self-modification and briefly introduced the advantages of this approach in relation to extended adaptability. This section is devoted to a deeper treatment of adaptability and principles of self-modification.

Adaptability is often considered in biological terms of natural evolution (Williams, 1996) or environmental uncertainty (Conrad, 1999) as well as in management and business processes (Gurvis & Calarco, 2007). There have been several attempts to create a common theory of adaptability, such as the approach suggested by Michael Conrad (Conrad, 1999). Overviewing the vast literature on the field of adaptation, we can recognize three main streams driving further development and representing different methodologies and different approaches to adaptation. The first and oldest stream is related to the theory of adaptive control. Several early works in adaptive control date from the late 50s - early 60s (Whitaker, 1959), (Osbourne *et al.*, 1961). In the mid-late 70s there appeared several issues related to temporary stabilities (Egardt, 1979), which in turn led to iterative control re-design and identification, and contributed in the mid-80s to robust adaptive control (Anderson *et al.*, 1986), (Rohrs *et al.*, 1985). Overviews of adaptive architectures can be found in textbooks (Narendra & Annaswamy, 1989), (Sastry & Bodson, 1989), which can be generalized as a high-level architecture, shown in Fig. 1.8(a) (Anderson, 2005).

Adaptive control consists of two parts, a conventional feedback-based control loop and adaptive part, depicted by the dashed line in Fig. 1.8(a). The environment is not explicitly integrated into this model, it is implicitly reflected by introducing disturbances and by uncertainties in the plant. The goal of the adaptive part is to estimate the behavior of a plant (by the identifier) and to calculate dynamically

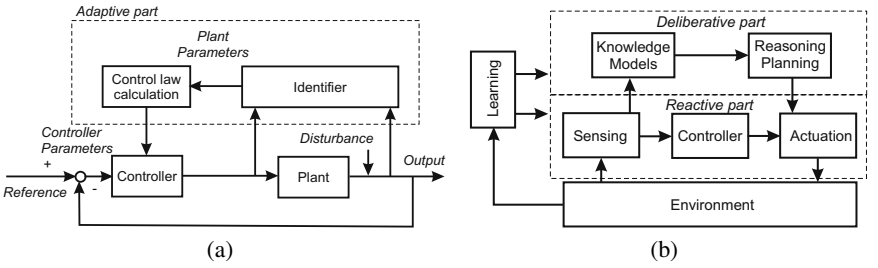


Fig. 1.8 A high-level architecture for **(a)** adaptive control **(b)** adaptive behavioral systems.

the control law (by the control law calculator). When in optimal control, a control law is designed off-line by a designer, an adaptive controller does it on-line. Most challenges in adaptive control theory are concentrated around adaptation of control to parameters of a plant when these parameters are unknown or changing.

The second mainstream of adaptation is located around adaptive behavior, which first arises within the AI community, e.g. (Beer, 1990), and involves cognitive aspects of adaption (Keijzer, 2003). There appear a few new components in the scheme from Fig. 1.8(a): explicit environment, sensing and actuation, as well as the deliberative cycle, shown in Fig. 1.8(b). When the reactive part of this scheme is in fact the optimal controller from Fig. 1.8(a), the deliberative part represents a new AI component. The adaptive system is now embedded into the unpredictable/dynamically changing environment; these systems are often referred to as situated systems (Mataric, 2002). Sensing and actuation represent a "body" of the system, intelligence (and so adaptation) is treated in term of embodiment (Pfeifer *et al.*, 2006). Achieving adaptivity in this context is spread into several approaches: different learning techniques in reactive and deliberative parts (Bull *et al.*, 2007), (Butz, 2002), (Puterman, 1994), behavior-based approaches (Kernbach *et al.*, 2009c), adaptive planning and reasoning (Weiss, 1999), biological inspiration in cognition (Cliff, 2003), evolutionary approaches (Alba & Tomassini, 2002) and many others. The goal of adaption can be formulated as achieving desired environmental responses according to some selected fitness/reward criteria.

The third mainstream towards adaptation is related to the community around distributed and software-intensive systems, computational, communication and sensor networks. With some degree of generalization, the business applications can be also related to this mainstream (SAP, 2005). The environment involves explicit users; the system itself is separated into different levels (applications), which run in parallel (Ledeczi *et al.*, 2000). The goal of adaptation here is related to scalability, self-optimization and self-protection, recognition of context, as well as to the software-engineering issues addressing reliability (Cheng *et al.*, 2008).

Based on this overview, we can say that adaptability represents a key point of systems working in real environments. Different uncertainties, variation of parameters

or even the appearance of unknown situations requires adaptive mechanisms, which allow the system to fit to these changed conditions. However, technical systems possess a goal-oriented behavior, they should be adaptive but also still be capable of achieving their design goals. To some extent, these systems are driven by two different forces: by fitness and by goal. In some cases, the goal of the system can also be focused on its own development. Here, the goal is transformed to the so-called self-concept and the system undergoes self-developmental processes. We now consider adaptive mechanisms and self-development in more detail.

Adaptation, Environment and Control. Since environmental changes require an adaptive reaction from a system, which in turn requires specific control mechanisms, we can divide changes and reactions into those forecast in advance and correspondingly those not forecast in advance. This division is relative, because in practical situations each change has forecasted and not forecasted components.

Adaptability is closely related to environmental changes and the ability of a system to react to these changes and the capability of the designer to forecast reaction of the environment to the system's response. Therefore adaptability is defined in term of the triple-relation: *environmental changes→ system's response → environmental reaction*. In general, adaptability is the ability of a collective system to achieve desired environmental reactions in accordance with a priori defined criteria by changing its own structure, functionality or behavior initiated by changed environment.

In Table 1.2 we roughly specify four different categories of environmental changes. According to environmental changes from this table, we can identify five

Table 1.2 Four types of environmental changes in robotic applications and examples of cases both forecast and not forecast in advance.

Environmental changes leading to:	Examples: Forecast in Advance	Examples: Not Forecast in Advance
Appearance of new situations	Installation of industrial robots in a new workshop	Work in previously unexplored environment (e.g. landing on Mars)
Changed functionality	Changing a type of locomotion (e.g. from wheeled to legged), when changing a terrain type	Search and rescue scenario when robots encounter unknown obstacles
Modified behavioral response	Gravitational perturbance of flying object in space and finding new control laws for engines	Distributed control of legged locomotion for obstacles of random geometry
Optimization of parameters	Changing of day/night light and adapting intensity of additional light	Adapting locomotive parameters for randomly moving obstacles

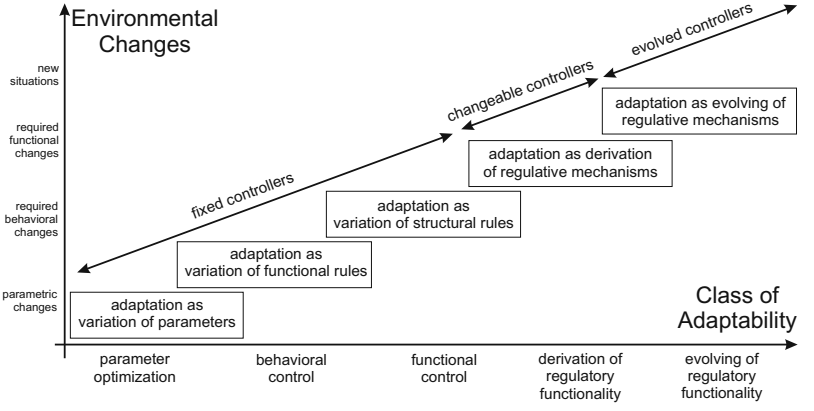


Fig. 1.9 Different adaptivity mechanisms in collective systems.

different classes of adaptability in collective systems, capable of structural phenomena: optimization mechanisms; behavioral control; functional control; derivation of new regulatory functionality and, finally, evolving of new regulatory functionality. These mechanisms are graphically represented in Fig. 1.9. To implement these adaptivity mechanisms we need to introduce two additional levels into the collective system from Fig. 1.6. The first level is related to control, we call it the *regulative level*, see Fig. 1.10. We find on this level different controllers, such as explicit and implicit rule-based (artificial neural networks), different bio-inspired, self-referred or learning systems.

These controllers influence structural or functional rules as well as change parameters of a corresponding level. All controllers work on the scheme *change of input parameters* $\rightarrow$ *changes of output parameters/rules*. The main goal of the regulative level is to maintain internal homeostasis of the system, to execute different tasks or, more generally, to demonstrate purposeful behavior depending on external conditions. Controllers at the regulative level allow some degree of adaptability for the system.

In detail, it depends to which extend a designer of these controllers was able to foresee possible changes of an environment and to integrate a reaction on these changes into controllers. The controllers allow different degrees of reaction on changes. However, the system at the regulative level is able to react only to changes whose parameter range was predicted in advance during the development of controllers or learning mechanisms. To react to such changes, which are not predictable at the design stage, we need to introduce a second level, which can modify regulative controllers – we denote this as the *generating level*. Following the scheme of adaptivity from Fig. 1.9, the generating level contains different deriving and evolving mechanisms.

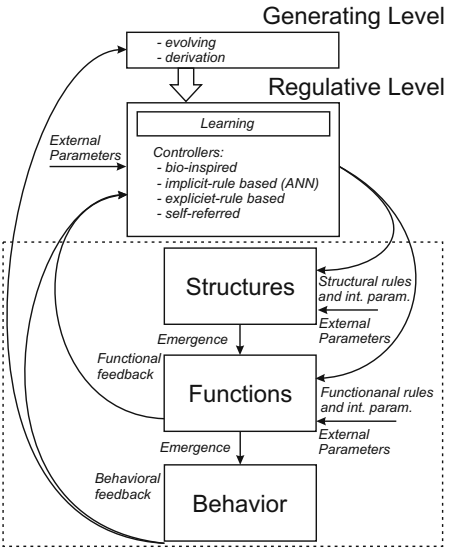


Fig. 1.10 Functional scheme of regulative and generating levels in structural systems.

Deriving is primarily related to distributed problem solving and planning approaches, known in the multi-agent community (Durfee, 1999), symbolic tasks decomposition (Kornienko *et al.*, 2004b), structural decomposition (Scassellati, 1998), self-referred dynamics (Kataoka & Kaneko, 2000) and others. These approaches are fast, deliver a predictable behavior and can be applied when a new situation is at least structurally known. Evolving is basically related to evolutionary approaches, see e.g. (Koza, 1992), and can be applied when the situation is completely unknown and a large search space of possible solutions should be explored. Recently, evolutionary approaches have been applied to a wide class of robotic problems (Floreano *et al.*, 2008a).

Invariant Goals, Self-Concept and Unbound Self-development. The mechanisms mentioned above allow adaptive behavior on different levels. However to avoid conflicts between achieving a goal and adapting to the environment, the goals at the generating level should be independent of adaptation processes, in other words, they should be formulated invariant to possible adaptations. There are several mechanisms expressing such an invariant property of the generating level: symmetries, conservation laws or e.g. "templates". Templates are well-known in cognitive science (Gobet & Simon, 1996) (also as "schemas" or "prototypes"), in topological research (in knot and braid theory) (Birman, 2004), as well as known as "frames" in the AI community (Minsky, 1977). The idea of a template is to describe most general "stereotypical" properties or features of some common class of situations/processes/objects.

As mentioned, goals can be focused on the system itself, i.e. they involve a self-concept. To explain the idea of the self-concept, we consider the case when a system should have a specific form, such as for the symmetric movement of legs, segmented (as in insects) construction of body, or there are imposed constraints or a priori desired properties. The self-concept contains in a compressed form a description of these "own" constraints or properties. The notion of self-concept originated in human psychological research, e.g. (McLean *et al.*, 2007), and is basically related to self-developmental processes. Self-development, is mostly known in human psychology, e.g. (Maslow, 1998), (McLean *et al.*, 2007), (Avstreih, 1981); in robotics the self-development usually refers to learning (Oudeyer & Kaplan, 2004) and

especially to ANN-based applications. Recently, there appear several works which apply psychological ideas to robotics, e.g. (Bonarini *et al.*, 2006), as well as the appearance of developmental robotics (Lungarella *et al.*, 2003) focused on ontogenetic processes related to cognitive science and the concept of embodiment.

Self-Development is bounded or unbounded process of functional, structural and regulatory changes undertaken by the system itself, related to its self-concept. A prerequisite is developmental plasticity on all levels. The self-concept can be expressed by symmetries, conservation laws, be planning- or fitness-driven or even have a character of unbounded metrics for open-ended evolution. Normally, self-development is initiated by differences between the self-concept and endogenous or environmental factors and may be unlimited in time and complexity.

In self-development we have to point out one principal element, related to the bounded and unbounded character of evolutionary changes. When in adaptive processes, these driving forces are mostly bounded, expressed by reward or fitness, the self-concept may include driving forces which are of unbounded character. In this way, self-development does not necessary imply any evolutionary progress, but a progress driven by the unbounded force of a self-concept. More generally, unbounded self-development (also denoted as open-ended evolution) *is characterized by a continued ability to invent new properties – so far only the evolution of life on Earth (data partly from the fossil record) and human technology (data from patents) have been shown to generate adaptive novelty in an open-ended manner* (Rasmussen *et al.*, 2004). We find some first ideas about open-ended evolution in (von Neumann, 1966) and (Waddington, 1969). Open-ended evolution is also related to indefinite growth of complexity (Ruiz-Mirazo *et al.*, 2008) and unbounded diversity (Maley, 1999). Ruiz-Mirazo and co-authors expressed the interesting idea that *"the combination of both self-assembly and self-organization processes within the same dynamic phenomenon can give rise to systems with increasing levels of molecular as well as organizational complexity"*. They also proposed to decouple genotype and phenotype from each other. A similar idea of increase homeostatic autonomy in macroevolution was proposed by (Rosslenbroich, 2009), which leads to us to not-fitness driven self-developmental processes. Several implementations of open-ended evolutionary scenarios, e.g. (Spector *et al.*, 2007), do not use any explicit behavioral fitness, moreover, there is no complexity growth in such "classical" artificial life simulator as Tierra and Avida (Standish, 2003). In this work Russell Standish proposed to improve these systems: "*a key step in doing this is to generate a process that adaptively recognises complexity, since it will be impossible to include humans in the loop, even when run on conventional computing platforms*".

These works lead us to an interesting question about the unbounded self-concept: which process can generate complexity? One of the first remarks is from von Neumann: "*synthesis of automata can proceed in such a manner that each automaton will produce other automata which are more complex and of*

Table 1.3 Several characteristics of self-developmental processes in collective systems.

Process	Developmental plasticity	Self-Concept
Regulatory	Structural and functional plasticity of the system, controllers can change their own transfer functions.	*(bound)* Achieving a targeted goal in changing environment. *(unbound)* Increasing performance characteristics.
Homeostatic	Like in the regulative case, but related to maintaining steady internal states in changing environment.	*(bounded)* Endogenous steady state. *(unbounded)* Achieving best possible homeostasis for diverse scalability metrics.
Learning	Changeable structure of regulative system.	*(bounded)* e.g. positive or negative rewards. *(unbounded)* Fitting very large (infinite) parameter space, e.g. by exploring structural-functional relations.
Planning-driven	Structural, functional and regulative plasticity.	*(bounded)* Minimizing deviations from a plan. *(unbounded)* Self-referred planning.
Fitness-driven	Structural, functional and regulative plasticity.	*(bounded)* Explicit fitness. *(unbounded)* Implicit fitness (optimizing energy balance, maximizing offsprings).
Open-ended	Capability for unbounded evolutionary activity.	*(unbounded)* Unbounded metrics.

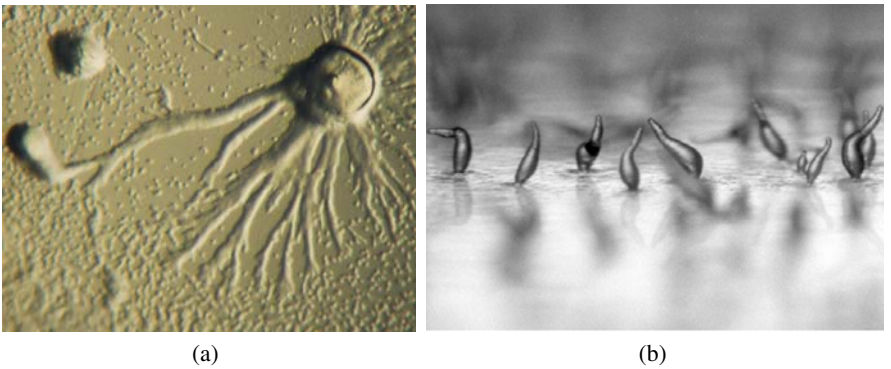
higher potentialities than itself" (von Neumann, 1966). A similar approach is observed in L-Systems (McCormack, 1993) (authors used evolutionary process but human operator in the selective loop) as well as in self-referred dynamics (Kataoka & Kaneko, 2000). It seems that structural production can lead to growth of complexity and diversity. However, considering the Kolmogorov complexity of fractal structures, which is equal to the shortest production set of rules (Kouptsov, 2008), we note the complexity of the whole fractal is independent of its size – the self-similar structural production does not increase complexity. Thus, we require that production systems include parameters which perturb generating structures. In this way, structural production rules parameterized by a random (environmental) value may lead to infinite growth of complexity and diversity, and are candidates for the unbounded self-concept. In Table 1.3 we collected several possible self-developmental processes in structural collective systems with bounded and unbounded self-concepts.

The final point which should be mentioned in this section is related to conflicts between achieving a goal and adaptive behavior. When a degree of adaptation is low, there are no essential conflicts between them. However, when plasticity is high, and the system can be hindered by adaptive processes from reaching the main goal, we are facing a new conceptual problem of a long-term controllability of adaptive and self-developmental processes. Obviously, either the goal should be formulated

in such an invariant way as allows multiple approaches for its achievements, or adaptive processes should basically be limited.

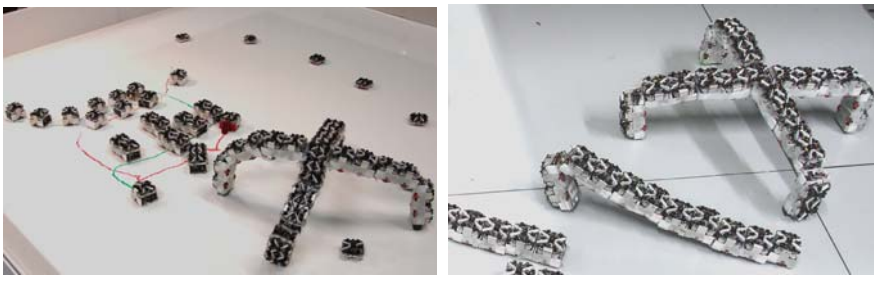
1.1.4 Artificial Symbiotic Systems: Perspectives and Challenges

As demonstrated in previous sections, collective systems capable of structural phenomena possess essential developmental plasticity, allow applications of horizontally and vertically self-organizing, planning- and fitness- based approaches and combine advantages provided by mono-functional and swarm-like systems. We can find in nature several examples of such systems, one of the most famous – Dictyostelium discoideum – social amoebae, known also as cellular slime molds (Kessin, 2001), see Fig. 1.11. These soil-living unicellular amoebae feed on bacteria. When the food resources run out, the amoebae produce and send signal molecules cAMP. This chemotaxis mechanism creates a gradient field towards an aggregation point and the collection of up to 100,000 cells first into a slug, see Fig. 1.11(a) and then into a fruiting body – a multi-cellular organism, Fig. 1.11(b). During this process, amoebae undergo different developmental processes such as cell differentiation, morphogenetic growth, self-protection, sexual and asexual reproduction and other. The principles of self-movement, aggregation and emergence of macroscopic functionality can be also demonstrated by artificial systems, in particular by swarm robots (Kornienko *et al.*, 2007). Like amoebae, swarm robots can send aggregation signals and aggregate into artificial organisms, see Fig. 1.12, and develop different macro-functionality through morphogenetic processes. In this way robots can combine collective mono-functional actuation with multi-functional collective phenomena, enabling advances in scalability and reliability. The research



(a) (b)

Fig. 1.11 Dictyostelium discoideum, commonly referred to as slime mold, capable of a transition from a collection of unicellular amoebae into a multicellular organism; **(a)** Dictyostelium Aggregation, (image source wikipedia); **(b)** Differentiation of unicellular amoeba Dictyostelium discoideum into multicellular "slugs" (100x) (Taken by Matthew Springer, University of California, San Francisco, using stereomicroscopy).



(a) (b)

Fig. 1.12 Swarm robots and a model of multi-robot organism; **(a)** Aggregation of swarm robot into several organisms; **(b)** Models of multi-robot organisms with different macro-actuation.

area of artificial organisms combines approaches and techniques from swarm, reconfigurable and evolutionary robotics.

Primarily, artificial organisms consist of heterogeneous modules. We distinguish between:

Active Module	Autonomous modules, capable of locomotion and actuation with different DoF. These modules possess independent energy source, computational and communication capabilities, see Sect. 2.1.
Passive Module	Non-autonomous modules, which do not have individual locomotion and actuation capabilities, however they are able to carry additional energy sources, structural load and other specialized passive functionality.
Tools	Autonomous modules, such as active wheels and grippers, which are specialized in some functionality. These modules also possess independent energy, computation and communication, see Sect. 2.1.6.

Following the idea of developmental plasticity, organisms are able to self-assemble and self-disassemble, see Sect. 4.5. In particular, this means that artificial organisms have two principally different states: swarm-mode and organism-mode, and progress through several phases: swarm, self-assembling, homeostatic regulation, macroscopic regulation, and so on, see Table 1.4. Organisms possess a homeostatic system, see Sect. 4.4, which performs self-regulation and energy management, common memory, common control, common event system, sensor-fusion and others, see Sects. 2.3.1.1 and 3.2. By changing scalability conditions, e.g. increasing the number of elements or diversity of elements, the self-regulatory systems try to establish an endogenous steady state. Finally, artificial organisms represent a combination of totally distributed and totally centralized regulatory systems. This is for the following reasons.

Table 1.4 Phases of the organism life cycle with respect to behavioral, functional, structural and self-developmental activities. Several phases can be executed simultaneously.

Phase	Behavioral, functional and structural activities	Self-developmental activities
Swarm	Massive-parallel execution of tasks typical for swarm: search, covering, resources localization and so on (**swarm-mode**).	Creating behavioral diversity by using population-based approaches, such as e.g. artificial sexuality and using functional self-organizing phenomena for behavioral emergence.
Self-assembling	Aggregation into planar structures, defining future topology and macroscopic functionality of an organism (**transition from swarm-mode to organism-mode**).	Diverse self-organizing, planning- and fitness-driven morphogenetic processes leading to functional emergence.
Homeostatic stabilization	Low- and intermediate-level processes, taking place immediately after mechanical self-assembly, and intended to regulate energetic, sensor, topological, information, memory and communication subsystems for the current topology of the organism (**organism-mode**).	Endogenous parametric self-regulation by using such approaches as artificial immune network, planning- or fitness-driven developmental mechanisms.
Cognitive regulation	Intermediate- and high-level processes of self-recognition (self-awareness) of own cognitive capabilities such as a number and functionality of available sensors, creating own- and world-models, creating sensomotor couplings and so on (**organism-mode**).	Creating and parameterizing internal cognitive structures and models basically by using planning- and fitness-driven mechanisms.
Macroscopic regulation	High-level processes, creating macroscopic regulative structure of the whole organism, related to macro-locomotion and macro-actuation, learning and evolving (**organism-mode**).	Main regulative self-developmental process creating control structures.
Self-repairing	Process leading to re-configuration or even removing of damaged modules from the organisms in case of any malfunctions. This is a relatively complex process based on self-diagnostic functionality of homeostatic subsystems and includes topology change phase and following homeostatic and regulative phases (**organism-mode**).	Self-developmental mechanisms are similar to homeostatic phase, however more strongly focused on self-diagnostics.
Macro-actuation	Working in the organism-mode to achieve targeted goals (**organism-mode**).	In this phase, organism performs learning and evolving activities related to behavioral and functional self-development.
Topology change	Taking a planar form for particular dissembling. This is a complex process, where all modules from old organism should not lose information collected in the organism mode. This leads to new homeostatic and regulative phases in a new organism (**organism-mode**).	This is the main structure-changing process, caused by learning, evolving or planning mechanisms decided to create new functionality and so a new structure.
Self-dis-assembling	Taking a planar form for a total dissembling (**transition from organism-mode to swarm-mode**).	Creating behavioral diversity as in the swarm phase.

By self-assembling and self-disassembling, organisms are working with different self-concepts. They may be small or large organisms with only active or passive elements, or they may have a combination of active, passive elements and tools. Thus, organisms should be able to work over a wide range of load and diverse scalability parameters. Obviously, strongly centralized regulatory systems have advantages for controlling complex macro-locomotion, however they have several difficulties with scalability and reconfigurability. Distributed swarm-like regulatory mechanisms have advantages in providing flexibility and scalability, however they have a relatively slow dynamics and can lead to communications overhead. We need a combination of central and distributed regulatory system which is fast enough for macro-locomotion but also flexible enough to deal with reconfigurability. This looks like the cooperative work of different sub-systems without building strongly centralized instances. Following bio-inspired ideas of cooperation between different species, we label these systems as symbiotic multi-robot organisms or artificial organisms.

Artificial Symbiotic Systems utilize structural developmental plasticity and a two layers regulatory architecture for creating a control system with horizontal operational principles. In particular, *symbiotic* means specific cooperation between different multi-functional regulative approaches, allowing a growth of hierarchies without building strongly centralized regulation for mono-functional collective activities.

Challenges and Perspectives. Issues of challenges in evolutionary, reconfigurable and swarm robotics have been described several times since the early 90s. We can refer to works (Mataric & Cliff, 1996), (Ficici *et al.*, 1999), (Lipson, 2000), (Sofge *et al.*, 2003), (Teo, 2004) related to challenges with fitness estimation, "reality gap" and others, whereas more recent works give overviews of challenges in robotics area (Siciliano & Khatib, 2008) such as over-motorization of reconfigurable systems or communication in swarm robotics. A combination of evolutionary techniques with other approaches creates new open questions about e.g. evolution and learning or "evolutionary self-organization" (where local rules are developed evolutionary but the self-organization remains outside of the evolutionary loop) in different scales. However, since artificial organisms combine all three areas, this results not only in a combination of problems and advantages, but also in the appearance of qualitatively new challenges. We believe that these new challenges are related to *developmental plasticity*, a *long-term independency* and to the *self-∗ issues*.

Developmental plasticity, as already discussed, in general means the structural and regulatory flexibility of basic elements, such as biological cells, hardware modules or software agents. The more plastic are basic elements, the more diverse and manifold are the resulting structures. Cells or bacteria provide great plasticity, whereas technological solutions are still far away from these biological solutions. Achieving a similar plasticity for artificial systems, by e.g. following inspirations from "natural chemistry", represents a serious challenge for the next few years.

Current robotic systems depend on maintenance, repair, specific energy sources and other infrastructure. These are not available in human-free environments or during long autonomous missions in e.g. space or ocean. A *long-term independence* means that robots can achieve their design goals without infrastructure and services provided by humans and in a variety of environments. This raises many different issues not only for the robot design, energy harvesting, reliability, adaptivity and regulatory autonomy, but also for predictability and controllability of long-term autonomy and self-development.

As already mentioned, swarm-like systems possess a high degree of redundancy and scalability. When some cell-modules in the organisms malfunction or are destroyed, they can be autonomously replaced by other cells, provided that some reserve of such cells exists. Thus, a combination of monofunctional actuation and swarm-like reliability may result in a new generation of self-monitoring and self-repairing systems. More generally, the *self-∗ issues*, i.e. self-awareness, self-reflection, self-regulation, self-reproductions, self-concept and others, are a sequence of different "self"-related problems. One of the main problems here consists not only in the lack of understanding of processes leading to e.g. self-reflection, self-awareness and consciousness, but also in the collective and emergent character of these phenomena. Thus, issues of plasticity, long-term independency and self-∗ problems are general challenges and, from this point of view, can be considered as the main benchmarks for artificial organisms.

Concluding this section, we would like to point out one important issue: artificial organisms can be viewed as extremely simplified analogues of living organisms. Both living and artificial organisms face similar problems – getting energy, surviving in the environment, different forms of self-protection and self-awareness, organization of long-term and short-term developmental processes and others. On the basis of artificial organisms we can gain deeper insights into a long-term evolution and its controllability, phenomena of individual and collective intelligence, mechanisms of multi-cellular regulation and other issues, which are of a great relevance in our understanding of the complexity of life.

1.2 Towards a Synergetic Quantum Field Theory for Evolutionary, Symbiotic Multi-Robotics

Paul Levi, Hermann Haken

We use the profound theoretical framework of the general quantum field theory (QFT), (Bjorken & Drell, 1965), (Penrose, 2006) to describe the interactions and dynamics of many-body systems which are in our case robot cells (or genes) developing an organism. Here it is important to state that we just use the formal analogy with QFT that is of purely formal nature, meaning that we operate here on a macroscopic level and no quantum effects appear or must be considered (Levi, 2009), (on the opposite to (Penrose, 1994)). In near future this assumption might be revised since NEMS (Nano-Electro-Mechanical-Systems) or even functional modules

that combine Nano-technology with molecules (NEBS, Nano-Electro-Biological-Systems) appear where quantum effects can not be excluded. The QFT-formalism is based on creation and annihilation operators that generate quantized field operators that in our contribution obey the non-relativistic QFT (Schrödinger field operators). On the opposite to the relativistic QFT these field operators can obey two different statistics (either Fermi-Dirac statistics or Bose-Einstein statistics), where these statistics are represented by two different types of commutator rules.

The operator $\mathbf{a}_j^\dagger(\underline{x}, t, s_j)$ creates a fermionic unit j (e.g. a robot-cell) at the position $\underline{x}$, at time t and the internal state s_j. In this contribution all operators will be written in bold letters. The operator $\mathbf{a}_j(\underline{x}, t, s_j)$ annihilates such a unit. It represents in mathematical terms the Hermitian conjugate operator. These two operators obey the exclusion principle of Pauli (e.g. no two robot-cells can take the same position and all other indices must also be different); therefore they anti-commutate (see appendix, Sect. 1.2.5.5): $\{\mathbf{a}, \mathbf{a}^\dagger\} = \mathbf{a}\mathbf{a}^\dagger + \mathbf{a}^\dagger\mathbf{a} = 1$. More generally spoken we consider every fermionic unit as an agent (Levi, 1989) that can directly interact with other agents or interact via exchanges of different messages with other agents. The notion of agent should clearly demonstrate our intention to include already on the level of robot-cells (or general to a fermionic units) cognitive abilities (Floreano & Mattiussi, 2008), (Trianni, 2008) that allow these units to aquire information and to perform also inferences that usually cannot be performed by units that are created by the classical QFT-theory.

In this formalism the direct interaction between different agents (on operator level) can be described by the following expression, where $\mathbf{O}(\underline{x}, \underline{x}')$ defines a position dependent operator (e.g. a transition operator):

$$\mathbf{a}_j^\dagger(\underline{x}, t, s_j)\, \mathbf{a}_k^\dagger(\underline{x}', t, s_k)\, \mathbf{O}(\underline{x}, \underline{x}')\, \mathbf{a}_k(\underline{x}', t, s_k)\, \mathbf{a}_j(\underline{x}, t, s_j). \tag{1.1}$$

The other class of operators are bosonic creation operators $\mathbf{b}^\dagger$ and annihilation operators $\mathbf{b}$. They obey the commutation relation $[\mathbf{b}, \mathbf{b}^\dagger] = \mathbf{b}\mathbf{b}^\dagger - \mathbf{b}^\dagger\mathbf{b} = 1$. These two operators obey the Bose-Einstein statistics and represent the fact that bosonic units can generate several fields that can be -in contrast to fermionic fields- in the same state (e.g. identical messages). In physics these operators describe all types of forces between elementary particles that are mediated by field operators $\Psi(x)$ and that are generated by the sum of corresponding particle operators. We use these operators to generate fields that can be seen as different types of message fields (e.g modes in the laser paradigm, (Haken, 1970)). In analogy to QFT we consider the generating units as different virtual particles that we call infermons.

An interaction of an agent (fermionic unit) with such a bosonic field is e.g. given by the following formula for the self-adjoint interaction Hamilton operator $\mathbf{H}_I$:

$$\mathbf{a}_j^\dagger(\underline{x}, t, s_l)\, \mathbf{a}_j(\underline{x}, t, s_m)\, \mathbf{b}^\dagger(\underline{x}, t) + \mathbf{a}_j^\dagger(\underline{x}, t, s_m)\, \mathbf{a}_j(\underline{x}, t, s_l)\, \mathbf{b}(\underline{x}, t). \tag{1.2}$$

The term $\mathbf{a}_j^\dagger(\underline{x}, t, s_l)\mathbf{a}_j(\underline{x}, t, s_m)$ describes the state transfer of a fermionic unit j from the energetic higher state s_m to a lower energetic state s_l by creation of the state s_l and the annihilation of the state s_m at the position $\underline{x}$ and the time t. As a result of

this operation the bosonic operator $\mathbf{b}^\dagger$ generates a field of a dedicated type (sending of a message) at the same position $\underline{x}$ and time t. The second term in this equation describes the process that a bosonic field is annihilated (unit j receives a message) and therefore annihilates the lower state s_l and transfers to the upper state s_m.

In the formalism of synergetics the generated bosonic "information fields" can effect as an order field (order "parameter" field) that synchronises the exchanges of different message types between robot-cells in order to generate coherent collective activities of these cells (Levi *et al.*, 1999). This coherence can also exist if the force mediating bosons are different. But it can also occur that the synchronisation of messages is not achievable (e.g. evident by a strong lumping of messages) resulting in a decoherence of the message flow. Such a field decoherence prevent the cooperation of cells. In term of synergetics this means that there is a threshold for order parameters. Beneath this threshold there is no coherency, at the threshold a bifurcation occurs, and above the threshold the heterogeneity can be controlled.

The QFT-formalisms in the deployment of an extended operator method delivers in addition equation of motions for each kind of creation operator and annihilation operator by the following commutator e.g. for $\mathbf{a}^\dagger$:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{a}^\dagger = \frac{\mathrm{i}}{\hbar}[\mathbf{H}_\mathrm{I}, \mathbf{a}^\dagger]. \tag{1.3}$$

A fundamental, undisturbed form of such an equation of motion for the creation operator $\mathbf{b}^\dagger$ is given by use of (1.5)

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{b}^\dagger = \mathrm{c}\mathbf{b}^\dagger - \mathrm{d}\mathbf{b}^\dagger(\mathbf{b}^\dagger\mathbf{b}). \tag{1.4}$$

This operational equation of motion, where c is a control parameter, d is proportional to a coupling parameter (often also incorrect named constant), and $\mathbf{b}^\dagger$ plays the role of an order operator field (order parameter field) is very characteristic for self-organising processes, since it represents the so-called Eigenanteile of such processes. This equation will build the basis of our approach and will be stepwise extended by perturbation and solved.

There is an additional rationale to us to involve QFT-formalism. This is the postulation that information is represented by a quantizing field operator whose components obey an equation of motion that is dictated by the interactions of both classes of operators (Haken, 2004).

As mentioned above we use fermionic operators $\mathbf{a}$ and $\mathbf{a}^\dagger$ for the construction and deconstruction of robot-cells (agents). In favour of genetic operations we use the following naming for bosonic operators: $\mathbf{cr}^\dagger$ (create cross-over) and $\mathbf{cr}$ (annihilate cross-over); $\mathbf{mt}^\dagger$ (create mutation) and $\mathbf{mt}$ (annihilate mutation). In addition we employ a more general bosonic message operator $\mathbf{ms}^\dagger$ (create a message), $\mathbf{ms}$ (annihilate a message) that is used to describe the temporal starting points and the duration of actions and can be applied to characterise conditions like temperature, pressure and concentrations under which e.g. the operations of cross-over and mutation have to take place. Here we just focus on the duration of an interaction that controls the on and off switching of individual operators. This process portrays the

selection of operations. We involve the operators $\mathbf{ms}^{\dagger}$ and **ms** to handle extrinsic and intrinsic sensor data to initiate and to terminate bosonic activities. But at last the sensor based selections result all time in start and stop instructions. Therefore we can describe with our "switch-model" all types of selection operations.

The activation respectively the deactivation of bosonic operators describe the effect of the agilities of regulation networks (Dayan & Abbott, 2001), (Alberts *et al.*, 2008). This means that we have separated and combined equations of motion for fermionic and bosonic operators, where the combined solutions can be considered as self-organised processes. Fermionic units generate bosonic fields and these exert again a feed-back "force" to the fermions. We abstain here to introduce additional genetic operators like $\mathbf{rp}^{\dagger}$ (creation of reproduction) and **rp** (destruction of reproduction), since the structure of the interactions of fermionic and bosonic units is in our case - comparable with the three body problem - already visible if we use three different types of mediating virtual bosonic particles (infermons).

In the construction of this chapter we are guided by the interaction of light and material (Haken, 1985). The model pattern is a laser that operates above the threshold. The quantum field approach approves in addition the beneficial separation of field equations and matter equations. This means that we can separate the equations of motion for the bosonic operators from the ones that are defined by e.g. for a flip operator $\mathbf{a}_{\mathrm{j}}^{\dagger}(\mathrm{m})\ \mathbf{a}_{\mathrm{j}}(\mathrm{l})$ for fermions with different internal states.

We summarise the introduction in order to give a road map for this contribution. Messages are represented by bosonic field quanta, they are generated and handled by agents (represented by fermionic operators that generate cognitive units). The message based interactions between agents generate the information that an agent can acquire. The complete approach is ambitious: QFT theory is fused with synergetics, and not enough, from this combination an unusual definition of information is deduced and distributed in a multi-agent system. It is intended that this information can be used for the description of the development of an artificial organism generated out from individual robot-cells. On this way we started our description of such a development by the development of water from H_2O molecules, since it is provable in reality. The final step do describe an organism as a many-body system where the component obey the Fermi-Dirac statistics (fermionic agents) and their interactions are mediated by different time dependent bosonic fields that represent e.g. external signals (**ms**) or different communications (**mt**, **cr**) between the components is still an ongoing work and is not finished in this contribution.

1.2.1 Cooperative (Coherent) Operations between Fermionic Units

1.2.1.1 Interaction Hamilton Operator

As it is usually done in physics we start with a Hamiltonian that describes the interaction between the fermionic operators and bosonic operators. The goal of this sub-paragraph is to present equations of motion for the three types of bosonic operators that result from their interactions with the fermionic units. This is important

since by synergetic principles the bosonic creation operators are considered as order "parameters" fields of the self-organisation process between agents and message fields (like the interaction of mater and light). Further we are looking for the control parameters that start such a self-organised process. To get these results we involve an adiabatic approach. Unfortunately these calculations are complex and long. Readers that are not accustomed to such operator calculus could consider this formalism as the way to deduce the elementary QFT-based self-organised equation that we mentioned already in the introduction (1.4).

Concerning the "agentifying" of the fermionic units this means that we have an overlay of multi-parameter fields where we have later on to analyse which parameter values primarily of the bosonic fields can guarantee the synchronisation (cooperation) of the agents by these fields.

We model the interactions between fermionic units in analogy to the resonant actions of a multimode laser in the representation of photonic statistics. This means in greater detail that if the energy E_m of state m is higher than the energy E_l of state l ($E_m - E_l > 0$) and this energy is equal e.g. to the energy $\hbar\omega_{jk}^{ms}$ of the bosonic field $\hbar\omega_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(n)\mathbf{ms}_{jk}(n)$ then cooperative (coherent) interactions between the fermionic agents can proceed. In this paper we work in the interaction representation and assume exact resonance, otherwise the Hamiltonians of the free fields must be taken into account. The interaction part of the Hamilton operator is defined by:

$$\begin{aligned}\mathbf{H}_I = {} & i\hbar \sum_{j,k,l,m,n} g_{jk}^{ms}\left(\mathbf{a}_j^{\dagger}(l)\mathbf{a}_j(m)\mathbf{ms}_{jk}^{\dagger}(n) - \mathbf{a}_j^{\dagger}(m)\mathbf{a}_j(l)\mathbf{ms}_{jk}(n)\right) + \\ & i\hbar \sum_{j,k,l,m,n} g_{jk}^{cr}\left(\mathbf{a}_j^{\dagger}(l)\mathbf{a}_j(m)\mathbf{cr}_{jk}^{\dagger}(n) - \mathbf{a}_j^{\dagger}(m)\mathbf{a}_j(l)\mathbf{cr}_{jk}(n)\right) + \\ & i\hbar \sum_{j,k,l,m,n} g_{jk}^{mt}\left(\mathbf{a}_j^{\dagger}(l)\mathbf{a}_j(m)\mathbf{mt}_{jk}^{\dagger}(n) - \mathbf{a}_j^{\dagger}(m)\mathbf{a}_j(l)\mathbf{mt}_{jk}(n)\right).\end{aligned} \tag{1.5}$$

The bosonic creation and annihilation operators generate or destroy a bosonic field by the interaction with fermionic fields. The indices have the following meaning: j is detached to fermionic units $\mathbf{a}_j^{\dagger}, \mathbf{a}_j$. The index k describes the message type (mode) of a bosonic field. The bosonic operators mediate the "forces" between the fermionic units and stand for the message exchange between them, where the more specific message type (bosonic states) are notated by n. The letters l and m describe internal states (e.g. exited state) of fermionic units. Bosonic operators that represent the interaction e.g. with the environment, or more general sensor data, are defined by $\mathbf{ms}_{jk}^{\dagger}(n)$ and $\mathbf{ms}_{jk}(n)$. The operators $\mathbf{cr}_{jk}$ and $\mathbf{cr}_{jk}^{\dagger}$; respectively $\mathbf{mt}_{jk}$ and $\mathbf{mt}_{jk}^{\dagger}$ define the direct internal interactions, e.g. the command for to genes to mutate.

The creation and annihilation operators are Hermetian adjoint, therefore complex coefficients should also be complex conjugated, only if they are real the coefficients for both conjugated operators are equal. Here we make the assumption that all coefficients are real.

The set of equations of motion for all three creation operators is given by (let n be fixed, and $\hbar = \frac{h}{2\pi}$ is Planck's quantum of action):

$$\frac{d}{dt}\mathbf{ms}^{\dagger}_{jk} = \frac{i}{\hbar}[\mathbf{H}_I, \mathbf{ms}^{\dagger}_{jk}] = -\frac{i}{\hbar}\frac{\partial \mathbf{H}_I}{\partial \mathbf{ms}_{jk}} = \sum_{l,m} g^{ms}_{jk}(\mathbf{a}^{\dagger}_j(m)\mathbf{a}_j(l)) = \sum_{l,m} g^{ms}_{jk}\boldsymbol{\alpha}^{\dagger}_j(m,l) \quad (1.6)$$

$$\frac{d}{dt}\mathbf{cr}^{\dagger}_{jk} = \frac{i}{\hbar}[\mathbf{H}_I, \mathbf{cr}^{\dagger}_{jk}] = -\frac{i}{\hbar}\frac{\partial \mathbf{H}_I}{\partial \mathbf{cr}_{jk}} = \sum_{l,m} g^{cr}_{jk}(\mathbf{a}^{\dagger}_j(m)\mathbf{a}_j(l)) = \sum_{l,m} g^{cr}_{jk}\boldsymbol{\alpha}^{\dagger}_j(m,l) \quad (1.7)$$

$$\frac{d}{dt}\mathbf{mt}^{\dagger}_{jk} = \frac{i}{\hbar}[\mathbf{H}_I, \mathbf{mt}^{\dagger}_{jk}] = -\frac{i}{\hbar}\frac{\partial \mathbf{H}_I}{\partial \mathbf{mt}_{jk}} = \sum_{l,m} g^{mt}_{jk}(\mathbf{a}^{\dagger}_j(m)\mathbf{a}_j(l)) = \sum_{l,m} g^{mt}_{jk}\boldsymbol{\alpha}^{\dagger}_j(m,l) \quad (1.8)$$

Here we abbreviated the formulas by the use of the state flip operator $\boldsymbol{\alpha}^{\dagger}_j(m,l) = \mathbf{a}^{\dagger}_j(m)\ \mathbf{a}_j(l)$. For further use we also introduce here in addition the Hermitean conjugated flip operator $\boldsymbol{\alpha}_j(m,l) = \mathbf{a}^{\dagger}_j(l)\ \mathbf{a}_j(m)$.

The second derivative of $\mathbf{ms}^{\dagger}_{jk}$ has the form (n is again fixed):

$$\frac{d^2}{dt^2}\mathbf{ms}^{\dagger}_{jk} = \sum_{l,m} g^{ms}_{jk}\frac{d}{dt}\boldsymbol{\alpha}^{\dagger}_j(m,l). \quad (1.9)$$

The derivation of the term $\boldsymbol{\alpha}^{\dagger}_j(m,l)$ is calculated by

$$\begin{aligned}\frac{d}{dt}\boldsymbol{\alpha}^{\dagger}_j(m,l) &= \frac{i}{\hbar}\left[\mathbf{H}_I, \boldsymbol{\alpha}_j(m,l)\right] \\ &= \sum_k g^{ms}_{jk}\mathbf{ms}^{\dagger}_{jk}\boldsymbol{\sigma}_j(m,l) + \sum_{k,l,m} g^{cr}_{jk}\mathbf{cr}^{\dagger}_j\boldsymbol{\sigma}_j(m,l) + \sum_{k,l,m} g^{mt}_{jk}\mathbf{mt}^{\dagger}_{jk}\boldsymbol{\sigma}_j(m,l).\end{aligned} \quad (1.10)$$

We used the following abbreviation for the self-conjugated operator

$$\boldsymbol{\sigma}_j(m,l) = \mathbf{a}^{\dagger}_j(m)\mathbf{a}_j(m) - \mathbf{a}^{\dagger}_j(l)\mathbf{a}_j(l) = \boldsymbol{\sigma}^{\dagger}_j(m,l). \quad (1.11)$$

In laser terminology $\boldsymbol{\sigma}_j$ describes the activity above the laser threshold (saturated inversion). In our case this denotes a time dependent activity of agent j where it produces messages (bosonic fields). This message generation can be quantified by the number of messages that are are transmitted (corresponds to the number of photons). We use the notation $\boldsymbol{\sigma}^0_j$ if agent j is in an equilibrium state where it generates no messages (typically activities below the laser threshold prescribing incoherent interactions with the surroundings).

The resulting formulas of the second derivations of the three creation operators are:

$$\frac{d^2}{dt^2}\mathbf{ms}^{\dagger}_{jk} = \sum_{l,m} g^{ms2}_{jk}\mathbf{ms}^{\dagger}_{jk}\boldsymbol{\sigma}(m,l). \quad (1.12)$$

$$\frac{d^2}{dt^2}\mathbf{cr}^{\dagger}_{jk} = \sum_{l,m} g^{cr2}_{jk}\mathbf{cr}^{\dagger}_{jk}\boldsymbol{\sigma}(m,l). \quad (1.13)$$

$$\frac{d^2}{dt^2}\mathbf{mt}_{jk}^{\dagger} = \sum_{l,m} g_{jk}^{mt2}\mathbf{mt}_{jk}^{\dagger}\boldsymbol{\sigma}(m,l). \tag{1.14}$$

The total sum of the "acceleration" of the creation operator $\mathbf{ms}_{jk}^{\dagger}$ is given by

$$\sum_{j,k}\frac{d^2}{dt^2}\mathbf{ms}_{jk}^{\dagger} = \sum_{j,k,l,m} g_{jk}^{ms2}\mathbf{ms}_{jk}^{\dagger}\boldsymbol{\sigma}(m,l). \tag{1.15}$$

Analogue formulas also hold for the two remaining creation operators. The influences of the three creation operators $\mathbf{ms}_{jk}^{\dagger}, \mathbf{cr}_{jk}^{\dagger}, \mathbf{mt}_{jk}^{\dagger}$ on the total interaction Hamiltonian are determined, beside the squared coupling constants $(g_{jk}^{ms})^2$, $(g_{jk}^{cr})^2$, $(g_{jk}^{mt})^2$, (all $\ll 1$), primarily by the rate of $\boldsymbol{\sigma}_j(m,l)$. Thus we calculate

$$\begin{aligned}\frac{d}{dt}\boldsymbol{\sigma}_j(m,l) = \frac{i}{\hbar}\left[\mathbf{H}_I, \boldsymbol{\sigma}_j^{\dagger}(m,l)\right] = -\Big(\sum_k g_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} + \sum_k g_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} + \\ \sum_k g_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}\Big)\boldsymbol{\alpha}_j(m,l) - \Big(\sum_k g_{jk}^{ms}\mathbf{ms}_{jk} + \sum_k g_{jk}^{cr}\mathbf{cr}_{jk} + \sum_k g_{jk}^{mt}\mathbf{mt}_{jk}\Big)\boldsymbol{\alpha}_j^{\dagger}(m,l).\end{aligned} \tag{1.16}$$

The commutator rules and the anti-commutator formulas must be valid for all time. This will be violated if we introduce only damping constants κ_{jk} (dissipation) for the creation operators, damping constants γ_{jk} for the flip operators, and a relaxation time T_j that defines the time span where $\boldsymbol{\sigma}_j$ recovers to the stationary value $\boldsymbol{\sigma}_j^0$. The validity of the commutator and anti-commutator formulas will be restored if we introduce external fluctuating forces F_{fluc}, which are mandatory that the commutators respectively the anti-commutators are exactly fulfilled all time (Haken, 1985). They represent the interaction with the environment. We will model these forces not explicitly in the interaction Hamiltonian (the formalism is simpler since we have not to calculate the expectation values of the stochastic forces) but implicitly with the aid of different statistics representing additional, stochastic interactions between the fields and the environmental restrictions (e.g. Poisson distribution for the mean number of messages).

If we include damping constants κ_{jk} into the formulas (1.6), (1.7) and (1.8) then the final set of equations is transferred to:

$$\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = \sum_{l,m} g_{jk}^{ms}\boldsymbol{\alpha}_j^{\dagger}(m,l) - \kappa_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}. \tag{1.17}$$

$$\frac{d}{dt}\mathbf{cr}_{jk}^{\dagger} = \sum_{l,m} g_{jk}^{cr}\boldsymbol{\alpha}_j^{\dagger}(m,l) - \kappa_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}. \tag{1.18}$$

$$\frac{d}{dt}\mathbf{mt}_{jk}^{\dagger} = \sum_{l,m} g_{jk}^{mt}\boldsymbol{\alpha}_j^{\dagger}(m,l) - \kappa_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}. \tag{1.19}$$

The temporal derivation of the flip operator is transferred to the following form if the damping constants γ_{jk} are included into formula (1.10):

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\alpha}_{\mathrm{j}}^{\dagger}(\mathrm{m},\mathrm{l}) = &\sum_{k}\left(\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) - \gamma_{\mathrm{jk}}^{\mathrm{ms}}\boldsymbol{\alpha}_{\mathrm{jk}}^{\dagger}(\mathrm{m},\mathrm{l})\right) + \\ &\sum_{k}\left(\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) - \gamma_{\mathrm{jk}}^{\mathrm{cr}}\boldsymbol{\alpha}_{\mathrm{jk}}^{\dagger}(\mathrm{m},\mathrm{l})\right) + \\ &\sum_{k}\left(\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) - \gamma_{\mathrm{jk}}^{\mathrm{mt}}\boldsymbol{\alpha}_{\mathrm{jk}}^{\dagger}(\mathrm{m},\mathrm{l})\right).\end{aligned} \tag{1.20}$$

In the last step we include the relaxation time $\mathrm{T_j(m,l)}$ in the formula (1.16). This step changes this equation into the following formula:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) = &-\left(\sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} + \sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} + \sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\right)\boldsymbol{\alpha}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) - \\ &\left(\sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} + \sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} + \sum_{k}\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\right)\boldsymbol{\alpha}_{\mathrm{j}}^{\dagger}(\mathrm{m},\mathrm{l}) + \\ &\frac{1}{\mathrm{T_j(m,l)}}\left(\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l}) - \boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\right).\end{aligned} \tag{1.21}$$

All in all we get a set of coupled non-linear differential equations. They can be solved if we assume that the damping constants and the relaxation time are different in the orders of magnitudes in the sense of adiabatic elimination (Haken, 1970). For example, for the "**ms**-constants" this estimation looks like

$$\gamma_{\mathrm{jk}}^{\mathrm{ms}} > \frac{1}{\mathrm{T_j(m,l)}} > \kappa_{\mathrm{jk}}^{\mathrm{ms}}. \tag{1.22}$$

Under the above mentioned assumption, that the damping constant γ_{jk} dominates the other parameters, we can set (if these inequalities are not true then a synchronised, self-organising process cannot start) $\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\alpha}_{\mathrm{j}}^{\dagger}(\mathrm{m},\mathrm{l}) = 0$ in (1.20) and solve this equation:

$$\boldsymbol{\alpha}_{\mathrm{j}}^{\dagger}(\mathrm{m},\mathrm{l}) = \sum_{\mathrm{k}}\left(\frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}}{\gamma_{\mathrm{jk}}^{\mathrm{ms}}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} + \sum_{\mathrm{k}}\frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}}{\gamma_{\mathrm{jk}}^{\mathrm{cr}}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} + \sum_{\mathrm{k}}\frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}}{\gamma_{\mathrm{jk}}^{\mathrm{mt}}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\right)\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}). \tag{1.23}$$

This result will then be inserted into (1.21) yielding the result:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) = -2\Big(&\sum_{\mathrm{k}}\frac{(\mathrm{g}_{\mathrm{jk'}}^{\mathrm{ms}})^2}{\gamma_{\mathrm{jk'}}^{\mathrm{ms}}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}} + \sum_{\mathrm{k}}\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}})^2}{\gamma_{\mathrm{jk}}^{\mathrm{cr}}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\mathbf{cr}_{\mathrm{jk}} + \\ &\sum_{\mathrm{k}}\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}})^2}{\gamma_{\mathrm{jk}}^{\mathrm{mt}}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\mathbf{mt}_{\mathrm{jk}}\Big)\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) + \frac{1}{\mathrm{T_j(m,l)}}\left(\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l}) - \boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\right).\end{aligned} \tag{1.24}$$

In the next step we set $\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l}) = 0$, solve the resulting equation with respect to $\boldsymbol{\sigma}_{\mathrm{j}}(\mathrm{m},\mathrm{l})$ and insert the result into (1.21). These steps generate the following approximate solution:

$$\begin{aligned}\boldsymbol{\alpha}_j^{\dagger}(m,l) = \boldsymbol{\sigma}_j^0(m,l)\sum_k \frac{g_{jk}^{ms}}{\gamma_{jk}^{ms}}\mathbf{ms}_{jk}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jk}^{ms})^2}{\gamma_{jk}^{ms}}\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}\right)+\\ \boldsymbol{\sigma}_j^0(m,l)\sum_k \frac{g_{jk}^{cr}}{\gamma_{jk}^{cr}}\mathbf{cr}_{jk}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jk}^{cr})^2}{\gamma_{jk}^{cr}}\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}\right)+\\ \boldsymbol{\sigma}_j^0(m,l)\sum_k \frac{g_{jk}^{mt}}{\gamma_{jk}^{mt}}\mathbf{mt}_{jk}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jk}^{mt})^2}{\gamma_{jk}^{mt}}\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}\right).\end{aligned} \tag{1.25}$$

In the last step we insert (1.24) e.g. into (1.16). The result for $\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger}$ reads then:

$$\begin{aligned}\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} &= sum_{l,m} g_{jk}^{ms}\boldsymbol{\alpha}_j^{\dagger}(m,l) - \kappa_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}\\ &= \sum_{l,m} g_{jk}^{ms}\boldsymbol{\sigma}_j^0(m,l)\Bigg(\sum_q \frac{g_{jq}^{ms}}{\gamma_{jq}^{ms}}\mathbf{ms}_{jq}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jq}^{ms})^2}{\gamma_{jq}^{ms}}\mathbf{ms}_{jq}^{\dagger}\mathbf{ms}_{jq}\right)+\\ &\quad \sum_q \frac{g_{jq}^{cr}}{\gamma_{jq}^{cr}}\mathbf{cr}_{jq}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jq}^{cr})^2}{\gamma_{jq}^{cr}}\mathbf{cr}_{jq}^{\dagger}\mathbf{cr}_{jq}\right)+\\ &\quad \sum_q \frac{g_{jq}^{mt}}{\gamma_{jq}^{mt}}\mathbf{mt}_{jq}^{\dagger}\left(1-2T_j(m,l)\frac{(g_{jq}^{mt})^2}{\gamma_{jq}^{mt}}\mathbf{mt}_{jq}^{\dagger}\mathbf{mt}_{jq}\right)\Bigg) - \kappa_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}.\end{aligned} \tag{1.26}$$

Finally, we rearrange this equation:

$$\begin{aligned}\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = &\sum_{l,m}\Bigg(g_{jk}^{ms}\boldsymbol{\sigma}_j^0(m,l)\Bigg(\sum_q \frac{g_{jq}^{ms}}{\gamma_{jq}^{ms}}\mathbf{ms}_{jq}^{\dagger}+\sum_q \frac{g_{jq}^{cr}}{\gamma_{jq}^{cr}}\mathbf{cr}_{jq}^{\dagger}+\sum_q \frac{g_{jq}^{mt}}{\gamma_{jq}^{mt}}\mathbf{mt}_{jq}^{\dagger}\Bigg)-\\ &\kappa_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\Bigg(\sum_q \frac{(g_{jq}^{ms})^3}{(\gamma_{jq}^{ms})^2}\mathbf{ms}_{jq}^{\dagger}\left(\mathbf{ms}_{jq}^{\dagger}\mathbf{ms}_{jq}\right)-\\ &\sum_q \frac{(g_{jq}^{cr})^3}{(\gamma_{jq}^{cr})^2}\mathbf{cr}_{jq}^{\dagger}\left(\mathbf{cr}_{jq}^{\dagger}\mathbf{cr}_{jq}\right) - \sum_q \frac{(g_{jq}^{mt})^3}{(\gamma_{jq}^{mt})^2}\mathbf{mt}_{jq}^{\dagger}\left(\mathbf{mt}_{jq}^{\dagger}\mathbf{mt}_{jq}\right)\Bigg)\Bigg).\end{aligned} \tag{1.27}$$

To generate the first derivative of the total bosonic field $\mathbf{ms}^{\dagger}$ (it summaries all created **ms**-messages) we must sum up over all indices j and k:

$$\begin{aligned}\frac{d}{dt}\mathbf{ms}^{\dagger} = &\sum_{l,m,j,k}\Bigg(g_{jk}^{ms}\boldsymbol{\sigma}_j^0(m,l)\Bigg(\sum_q \frac{g_{jq}^{ms}}{\gamma_{jq}^{ms}}\mathbf{ms}_{jq}^{\dagger}+\sum_q \frac{g_{jq}^{cr}}{\gamma_{jq}^{cr}}\mathbf{cr}_{jq}^{\dagger}+\sum_q \frac{g_{jq}^{mt}}{\gamma_{jq}^{mt}}\mathbf{mt}_{jq}^{\dagger}\Bigg)-\\ &\kappa_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\Bigg(\sum_q \frac{(g_{jq}^{ms})^3}{(\gamma_{jq}^{ms})^2}\mathbf{ms}_{jq}^{\dagger}\left(\mathbf{ms}_{jq}^{\dagger}\mathbf{ms}_{jq}\right)-\\ &\sum_q \frac{(g_{jq}^{cr})^3}{(\gamma_{jq}^{cr})^2}\mathbf{cr}_{jq}^{\dagger}\left(\mathbf{cr}_{jq}^{\dagger}\mathbf{cr}_{jq}\right) - \sum_q \frac{(g_{jq}^{mt})^3}{(\gamma_{jq}^{mt})^2}\mathbf{mt}_{jq}^{\dagger}\left(\mathbf{mt}_{jq}^{\dagger}\mathbf{mt}_{jq}\right)\Bigg)\Bigg).\end{aligned} \tag{1.28}$$

Analogous equations are true for $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{cr}^{\dagger}$ and $\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{mt}^{\dagger}$. We have now defined all formulas and can begin to solve them. The first step in this direction will be the calculation of the solution of (1.26). Afterwards the analogous equations for the temporal derivations of $\mathbf{cr}_{\mathrm{jk}}^{\dagger}$ and $\mathbf{mt}_{\mathrm{jk}}^{\dagger}$ must be solved.

We perform this procedure step by step, and we start with the simplification of Eq. (1.26). This yields the formula:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} = \boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\left(\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}})^{2}}{\gamma_{\mathrm{jk}}^{\mathrm{ms}}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} - \kappa_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} + \frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}}{\gamma_{\mathrm{jk}}^{\mathrm{cr}}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} + \frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}}{\gamma_{\mathrm{jk}}^{\mathrm{mt}}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\right) - \\ 2\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\mathrm{T}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\left(\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{ms}})^{2}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\left(\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jq}}\right) + \frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{cr}})^{2}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\left(\mathbf{cr}_{\mathrm{jk}}^{\dagger}\mathbf{cr}_{\mathrm{jk}}\right) + \right. \\ \left.\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{mt}})^{2}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\left(\mathbf{mt}_{\mathrm{jk}}^{\dagger}\mathbf{mt}_{\mathrm{jk}}\right)\right).\end{aligned} \tag{1.29}$$

In short:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} = \mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} - \mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\left(\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}\right) + \mathrm{c}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} - \mathrm{d}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\left(\mathbf{cr}_{\mathrm{jk}}^{\dagger}\mathbf{cr}_{\mathrm{jk}}\right) + \\ \mathrm{c}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger} - \mathrm{d}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\left(\mathbf{mt}_{\mathrm{jk}}^{\dagger}\mathbf{mt}_{\mathrm{jk}}\right).\end{aligned} \tag{1.30}$$

We introduced the following abbreviations:

$$\begin{aligned}&\mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}} = \boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}})^{2}}{\gamma_{\mathrm{jk}}^{\mathrm{ms}}} - \kappa_{\mathrm{jk}}^{\mathrm{ms}}\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l}), \quad \mathrm{c}_{\mathrm{jk}}^{\mathrm{cr}} = \boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}}}{\gamma_{\mathrm{jk}}^{\mathrm{cr}}}, \\ &\mathrm{c}_{\mathrm{jk}}^{\mathrm{mt}} = \boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\frac{\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}}\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}}}{\gamma_{\mathrm{jk}}^{\mathrm{mt}}}; \\ &\mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}} = 2\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\mathrm{T}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{ms}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{ms}})^{2}}, \quad \mathrm{d}_{\mathrm{jk}}^{\mathrm{cr}} = 2\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\mathrm{T}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{cr}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{cr}})^{2}}, \\ &\mathrm{d}_{\mathrm{jk}}^{\mathrm{mt}} = 2\boldsymbol{\sigma}_{\mathrm{j}}^{0}(\mathrm{m},\mathrm{l})\mathrm{T}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\frac{(\mathrm{g}_{\mathrm{jk}}^{\mathrm{mt}})^{3}}{(\gamma_{\mathrm{jk}}^{\mathrm{mt}})^{2}}.\end{aligned}$$

In analogy to (1.30) we rewrite the equation for $\mathbf{cr}_{\mathrm{jk}}^{\dagger}$:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} = \mathrm{e}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger} - \mathrm{f}_{\mathrm{jk}}^{\mathrm{cr}}\mathbf{cr}_{\mathrm{jk}}^{\dagger}\left(\mathbf{cr}_{\mathrm{jk}}^{\dagger}\mathbf{cr}_{\mathrm{jk}}\right) + \mathrm{e}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} - \mathrm{f}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\left(\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}\right) + \\ \mathrm{e}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger} - \mathrm{f}_{\mathrm{jk}}^{\mathrm{mt}}\mathbf{mt}_{\mathrm{jk}}^{\dagger}\left(\mathbf{mt}_{\mathrm{jk}}^{\dagger}\mathbf{mt}_{\mathrm{jk}}\right).\end{aligned} \tag{1.31}$$

Similar abbreviations are:

$$e_{jk}^{ms} = \boldsymbol{\sigma}_j^0(m,l)\frac{(g_{jk}^{cr})^2}{\gamma_{jk}^{cr}} - \kappa_{jk}^{cr}\boldsymbol{\sigma}_j^0(m,l), \quad e_{jk}^{ms} = \boldsymbol{\sigma}_j^0(m,l)\frac{g_{jk}^{cr}g_{jk}^{ms}}{\gamma_{jk}^{ms}},$$

$$e_{jk}^{mt} = \boldsymbol{\sigma}_j^0(m,l)\frac{g_{jk}^{cr}g_{jk}^{mt}}{\gamma_{jk}^{mt}};$$

$$f_{jk}^{cr} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{cr})^3}{(\gamma_{jk}^{cr})^2}, \quad f_{jk}^{ms} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{ms})^3}{(\gamma_{jk}^{ms})^2},$$

$$f_{jk}^{mt} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{mt})^3}{(\gamma_{jk}^{mt})^2}.$$

In the end the analogous equation for $\mathbf{mt}_{jk}^{\dagger}$ looks like:

$$\frac{d}{dt}\mathbf{mt}_{jk}^{\dagger} = s_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger} - t_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}) + s_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - t_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + s_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} - t_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}). \tag{1.32}$$

The abbreviations are again conforming to the previous one:

$$s_{jk}^{mt} = \boldsymbol{\sigma}_j^0(m,l)\frac{(g_{jk}^{mt})^2}{\gamma_{jk}^{mt}} - \kappa_{jk}^{mt}\boldsymbol{\sigma}_j^0(m,l), \quad s_{jk}^{ms} = \boldsymbol{\sigma}_j^0(m,l)\frac{g_{jk}^{mt}g_{jk}^{ms}}{\gamma_{jk}^{mt}},$$

$$s_{jk}^{cr} = \boldsymbol{\sigma}_j^0(m,l)\frac{g_{jk}^{cr}g_{jk}^{mt}}{\gamma_{jk}^{mt}};$$

$$t_{jk}^{mt} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{mt})^3}{(\gamma_{jk}^{mt})^2}, \quad t_{jk}^{ms} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{ms})^3}{(\gamma_{jk}^{ms})^2},$$

$$t_{jk}^{cr} = 2\boldsymbol{\sigma}_j^0(m,l)T_j(m,l)\frac{(g_{jk}^{cr})^3}{(\gamma_{jk}^{cr})^2}.$$

Eqs. (1.30) - (1.32) and their three Hermitean conjugates are the differential equations we have to solve. All these three differential equations are composed of three symmetrical parts. Nevertheless we cannot solve each symmetrical term by itself and then add all three together because the full equations are non-linear. In addition the actions of these three parts e.g. of (1.30) are different. The first term models the own contribution (Eigenanteil) $\mathbf{ms}_{jk}^{\dagger}$, the remaining two parts describe the coupling of $\mathbf{ms}_{jk}^{\dagger}$ to the creation operators $\mathbf{cr}_{jk}^{\dagger}$ and $\mathbf{mt}_{jk}^{\dagger}$. The mutual interactions of all three bosonic fields denotes a strong dependency between them, meaning e.g. that the **ms**-field tries to control the **mt**-field, or by interference effects new combinations of frequencies occur, or the frequency of two fields is changed. In technical terms this means that there exist frequency lockings and phase lockings that are generated by the interactions of these three modes.

Transferred to symbiotic organisms these effects can stabilise e.g. the connections of different cells since there is a strong regulative, dynamic regime that controls the internal communications by **mt**-fields and **cr**-fields by the **ms**-field. But it is also possible (depending from the settings of the various parameters) that the **mt**-field (mutation) dominates the other two fields and the organism destabilises. The kind of interactions of the three bosonic fields define whether the robot-cells can cooperate together. In a synergetic view all this depends on the values of the control parameters e.g. c_{jk}^{ms} of the **ms**-field and the dynamics of the order field $\mathbf{ms}_{jk}^{\dagger}$, (see next sub-paragraph).

The additional principal tools beside the above performed synergetic based calculations (e.g. control parameter, order parameter field (Haken, 1977) are the methods of dynamical systems (Ghrist *et al.*, 1997) and the techniques of differential geometry (Kobayashi & Nomizu, 1996), (Guckenheimer & Holmes, 1983). We start the descriptions of the solutions by the view of dynamical systems. Hereby, as a standard procedure, the equilibrium of a dynamical system will be studied by the behavior of an invariant set Λ of the vector field $f(\Lambda) = \Lambda$. Such a set Λ can be one-point set (fixed-point) or a manifold (e.g. circle of fixed points, see Fig. 1.13(b)). In any case, we analyse the behavior of a dynamical system primarily in equilibrium states in order to describe the stability of the system invariant sets.

1.2.2 Individual Contributions of the Eigenanteile

In the first step the equations of the bosonic creation operators $\mathbf{ms}_{jk}^{\dagger}$ and $\mathbf{ms}_{jk}$ are solved. Since we are interested in the results of the measurements of these two operators we calculate the expectation values of them. It is also usual to analyse the flow of the field. We do this in the second sub-chapter. In the third sub-chapter an external, periodic force affects the expectation values of the two **ms**-operators, where such a field e.g. come from a periodic electro-magnetic wave. If we go deeper into the physics such a wave can polarise the robot cells. The calculated phase portraits demonstrate the resulting periodicity in this representation space.

1.2.2.1 Uncoupled and Unforced Contributions of the Operators and Their Expectation Values

In order to get an impression of the first incomplete results of (1.30) we study the behavior of the Eigenanteile of $\mathbf{ms}_{jk}^{\dagger}$ (Eigenanteil-results for $\mathbf{cr}_{jk}^{\dagger}$ and $\mathbf{mt}_{jk}^{\dagger}$ are similar). Such Eigenanteile represent in a self consistent way how the "**ms**-field" interact with itself via the interaction with a fermionic agent. In more detail; we separate from (1.30) the partial formula

$$\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = c_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}). \tag{1.33}$$

Supplementary we write down the conjugate equation

$$\frac{d}{dt}\mathbf{ms}_{jk} = c_{jk}^{ms}\mathbf{ms}_{jk} - d_{jk}^{ms}\mathbf{ms}_{jk}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}). \tag{1.34}$$

These two formulas are the first elementary building blocks of our approach. The operator $\mathbf{ms}_{jk}^{\dagger}$ plays the role of an order parameter, c_{jk}^{ms} defines the control parameter, and formula (1.25) expresses the slaving principle.

The creation respectively annihilation operators are complex operators. The appropriate method of solutions of the two adjoint operator equations can be done in the space of eigenfunctions Φ_α of the annihilation operator $\mathbf{ms}_{jk}$, where

$$\mathbf{ms}_{jk}\Phi_\alpha = \alpha\Phi_\alpha, \alpha \in \mathbb{C}. \tag{1.35}$$

We take the expectation value

$$\left\langle \Phi_\alpha \left| \mathbf{ms}_{jk} \right| \Phi_\alpha \right\rangle = \alpha, \tag{1.36}$$

with the normalisation $\langle \Phi_\alpha \mid \Phi_\alpha \rangle = 1$. For the adjoint operator holds $\langle \phi_\alpha \mid \mathbf{ms}_{jk}^{\dagger} \mid \phi_\alpha \rangle = \alpha^*$. Below we will use the short notations $\langle \mathbf{ms}_{jk} \rangle$ and $\langle \mathbf{ms}_{jk}^{\dagger} \rangle$ for these two expectation values.

For clarity we drop the indices in the two operator equations (1.33) and (1.34) and get the equivalent equations for the expectation values

$$\frac{d\alpha}{dt} = c\alpha - d|\alpha|^2\alpha, \ \frac{d\alpha^*}{dt} = c\alpha^* - d|\alpha|^2\alpha^*. \tag{1.37}$$

If we split the expectation values of the **ms** operator into the real part $u = \mathrm{Re}\left\langle \mathbf{ms}_{jk} \right\rangle$ and imaginary part $v = \mathrm{Im}\left\langle \mathbf{ms}_{jk} \right\rangle$ then we obtain from both equations the following differential equations for u and v:

$$\frac{d}{dt}u = c_{jk}^{ms}u - d_{jk}^{ms}u(u^2+v^2) = c_{jk}^{ms}u - d_{jk}^{ms}ur^2. \tag{1.38}$$

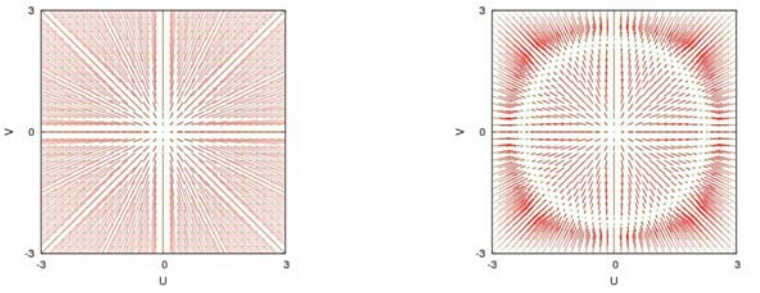
$$\frac{d}{dt}v = c_{jk}^{ms}v - d_{jk}^{ms}v(u^2+v^2) = c_{jk}^{ms}v - d_{jk}^{ms}vr^2. \tag{1.39}$$

These equations are symmetric in (u,v). The single unstable fixed point is the origin $O = (0,0)$, if $c_{jk}^{ms} > 0$ and $d_{jk}^{ms} < 0$. It represents a saddle point that is physically not relevant. Fig. 1.13(a) demonstrates this fact by a phase portrait of u and v, and it shows the influence of the unstable fixed point in the origin O[1].

The alternative calculations with $\langle \mathbf{ms}_{jk} \rangle = \alpha = r(t)e^{i\varphi(t)}$ and $\langle \mathbf{ms}_{jk}^{\dagger} \rangle = \alpha^*$ deliver in a more elegant way the two equivalent differential equations if both coefficients are real:

$$\frac{d}{dt}r = c_{jk}^{ms}r - d_{jk}^{ms}r^3, \text{and} \frac{d}{dt}\varphi = 0. \tag{1.40}$$

[1] We express our gratitude to Dr. Victor Avrutin for his uncomplaining and substantial calculation support of this work.



(a) (b)

Fig. 1.13 Phase portraits of u and v. **(a)** There is an unstable fixed point in the origin origin O = (0,0). The values are $c_{jk}^{ms} = 1.5$ and $d_{jk}^{ms} = -0.3$; **(b)** The values are $c_{jk}^{ms} = 1.5$ and $d_{jk}^{ms} = 0.3$.

Fig. 1.13(b) demonstrates the effect if d_{jk}^{ms} gets positive; there arises a circle of attractive fixed points with radius $r_0 = \sqrt{\frac{c_{jk}^{ms}}{d_{jk}^{ms}}}$. For physical relevant "laser actions" we have to use a positive d_{jk}^{ms} in order to get a stable amplitude.

1.2.2.2 Flow of the Continuous and Uncoupled Eigenanteile

Here we specify the explicit solutions of the two Eqs. 1.33 and 1.34, and calculate the accordant flow in terms of dynamical systems. A dynamical system $\frac{dx}{dt} = f(x)$, is a system of differential equations, where x = x(t)$\in \mathrm{R}^n$ and f is a vector field that generates a continuous flow $\Phi_t(x) = \Phi(x,t)$, that also can be considered as a one parameter group under the operation of composition $(\Phi_s \circ \Phi_t(x)) = \Phi_{s+t}(x)$ that satisfies the equation

$$\frac{d}{dt}\Phi(x,t)|_{t=\tau} = f(\Phi(x,\tau)), \forall\ x \text{ and } \tau \in I = (a,b) \subseteq \mathbb{R}^n. \tag{1.41}$$

We repeat again the continuous Eigenanteil

$$\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = c_{j,k}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{j,k}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}). \tag{1.42}$$

If we drop all indices of (1.37) then we get the solutions:

$$r(t) = \sqrt{c}\Big((\frac{c}{r_0^2} - d)e^{-2ct} + d\Big)^{-1/2},\ \varphi = \varphi_0. \tag{1.43}$$

The fixed point is, as we already know, defined by $r_0 = \sqrt{\frac{c}{d}}$; its stability is calculated by

$$\frac{dr}{dr_0}|_{\sqrt{\frac{c}{d}}} = de^{-2ct}. \tag{1.44}$$

This fixed point r_0 is stable, if c and d are both positive, reflecting the already known circular orbit of radius r_0. In the limit $t \to \infty$, $r(t)$ converges to r_0. We get an equivalent result if we consider the derivative D of the Poincaré map

$$DP(r_0) = \frac{dP}{dr_0}|_{\sqrt{\frac{c}{d}}} = d^{3/2}e^{-4\pi c} < 1. \tag{1.45}$$

The continuous flow operator is given by:

$$\Phi_t(r_0, \varphi_0) = \left(\sqrt{c}\left(\left(\frac{c}{r_0^2} - d\right)e^{-2ct} + d\right)^{-1/2}, \varphi_0\right). \tag{1.46}$$

Fig. 1.14 demonstrates the flow (one-parameter group) for $r(t)$ and different constant φ values.

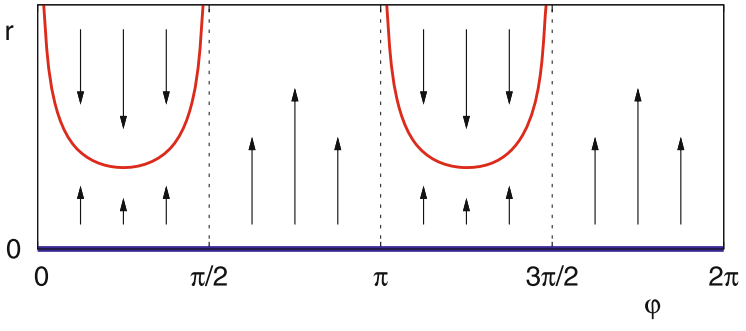


Fig. 1.14 Schematic representation of flow (phase curves) in the (r, φ) plane with $r(\varphi) = \sqrt{\frac{c}{d}}$. There is a pattern repetition for $0 < \varphi < \frac{\pi}{2}$, $\pi < \varphi < \frac{3\pi}{2}$, etc. Red curves are attractive; the blue line is repulsive (unstable fixed point in the origin).

1.2.2.3 Uncoupled and External Forced Oscillations of the Operators and Their Expectation Values

In the next step we apply an external periodic force $A\sin(\omega t)$ with real amplitude A:

$$\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = c_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + A\sin(\omega t). \tag{1.47}$$

$$\frac{d}{dt}\mathbf{ms}_{jk} = c_{jk}^{ms}\mathbf{ms}_{jk} - d_{jk}^{ms}\mathbf{ms}_{jk}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + A\sin(\omega t). \tag{1.48}$$

In the short notation – with coefficient dropping – these two equations read as $(\alpha = r(t)e^{i\varphi(t)})$

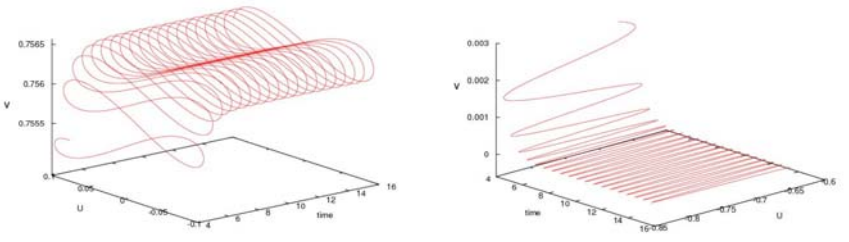
$$\frac{\mathrm{d}}{\mathrm{dt}}\alpha = \mathrm{c}\alpha - \mathrm{d}|\alpha|^2\alpha + \mathrm{A}\sin(\omega \mathrm{t}),\ \frac{\mathrm{d}}{\mathrm{dt}}\alpha^* = \mathrm{c}\alpha^* - \mathrm{d}|\alpha|^2\alpha^* + \mathrm{A}\sin(\omega \mathrm{t}). \tag{1.49}$$

If we take again for granted that c and d are real then the resulting differential equations are $(\alpha = \mathrm{r}(\mathrm{t})\mathrm{e}^{\mathrm{i}\varphi(\mathrm{t})})$:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{r} = \mathrm{cr} - \mathrm{dr}^3 + \mathrm{A}\sin(\omega \mathrm{t})\cos(\varphi). \tag{1.50}$$

$$\mathrm{r}\frac{\mathrm{d}}{\mathrm{dt}}\varphi = \mathrm{A}\sin(\omega \mathrm{t})\sin(\varphi). \tag{1.51}$$

Represented in the Cartesian coordinates (u, v) two different solutions are demonstrated in Fig. 1.15. In Fig. 1.15(a) the solution has a form of an eight that twist up; in Fig. 1.15(b) the solution twists down to the plane $\mathrm{v} = 0$, where it performs sinus oscillations.



(a) (b)

Fig. 1.15 Phase portrait of $\alpha = \mathrm{u} + \mathrm{iv}$.The parameter values are: **(a)** $\mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}} = -0.75$, $\mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}} = 1.3$, A = 1, $\omega = 10$; **(b)** $\mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}} = 0.75$, $\mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}} = 1.3$, A = 1, $\omega = 10$.

1.2.3 Separate Perturbations of the Eigenanteile

In the next step we disturb the Eigenanteile e.g. for $\mathbf{ms}_{\mathrm{jk}}^{\dagger}$(similar results are obtained also for the other Eigenanteil-equations) by a mixed term $\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}$:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} &= \mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger} - \mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}(\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}} \\
\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{\mathrm{jk}} &= \mathrm{c}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}} - \mathrm{d}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}(\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}_{\mathrm{jk}}^{\mathrm{ms}}\mathbf{ms}_{\mathrm{jk}}^{\dagger}\mathbf{ms}_{\mathrm{jk}}.
\end{aligned} \tag{1.52}$$

A similar procedure as before, $\mathbf{ms}_{\mathrm{jk}} = \mathrm{r}(\mathrm{t})\mathrm{e}^{\mathrm{i}\varphi(\mathrm{t})}$, $\mathbf{ms}_{\mathrm{jk}}^{\dagger} = \mathrm{r}(\mathrm{t})\mathrm{e}^{-\mathrm{i}\varphi(\mathrm{t})}$, transfers Eq. (1.50) with dropped indices into

$$\mathrm{i}(\frac{\mathrm{d}}{\mathrm{dt}}\varphi)\mathrm{r} + \frac{\mathrm{d}}{\mathrm{dt}}\mathrm{r} = \mathrm{cr} - \mathrm{dr}^3 + \hat{\mathrm{g}}\mathrm{r}^2\mathrm{e}^{-\mathrm{i}\varphi}. \tag{1.53}$$

The splitting of this formula into real part and imaginary part generates the result

$$\frac{d}{dt}r = cr - dr^3 + \hat{g}r^2\cos(\varphi). \tag{1.54}$$

$$\frac{d}{dt}\varphi = -\hat{g}r\sin(\varphi),\ (r \neq 0). \tag{1.55}$$

There is an unstable fixed point for $r_0 = 0$ and φ arbitrary; and a stable fixed point in

$$r_0 = \frac{1}{2d}(\hat{g} + \sqrt{\hat{g}^2 + 4cd}) \text{ and } \sin(\varphi_0) = 0,\ \hat{g} \neq 0. \tag{1.56}$$

In Cartesian coordinates Eqs. (1.52) read as:

$$\frac{d}{dt}u = c_{jk}^{ms}u - d_{jk}^{ms}u(u^2 + v^2) + \hat{g}_{jk}^{ms}u(u^2 + v^2) = c_{jk}^{ms}u - d_{jk}^{ms}ur^2 + \hat{g}_{jk}^{ms}ur^2. \tag{1.57}$$

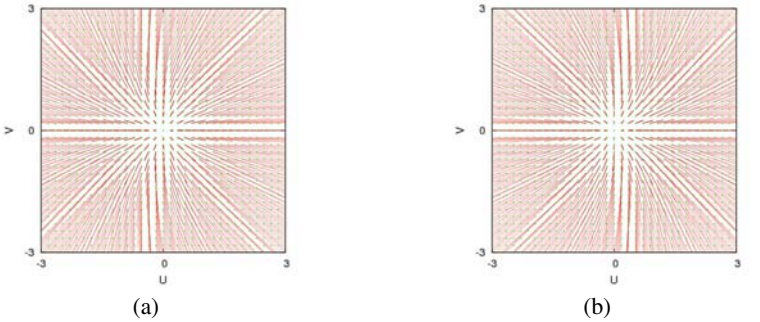
$$\frac{d}{dt}v = c_{jk}^{ms}v - d_{jk}^{ms}v(u^2 + v^2) = c_{jk}^{ms}v - d_{jk}^{ms}vr^2. \tag{1.58}$$

The nontrivial fixed points in the (u, v) space are:

$$u_0 = \pm r_0, v_0 = 0. \tag{1.59}$$

Fig. 1.16 shows the phase flow of these two equations with respect to the unstable fixed point in the origin. By comparison with Fig. 1.13(a) we see that the flow outside the u axis turns to the right side (a) or left side (b).

In the next parameter fixation c_{jk}^{ms} stays unchanged and d_{jk}^{ms} gets positive. Fig. 1.17 demonstrates the definite change of the flow behavior. In Fig. 1.17(a) the fixed point in the origin stays unstable, in addition two new fixed points $u_0 = \pm r_0$ are created. The right fixed point $u_0 = r_0$ is stable and the left fixed point $u_0 = -r_0$ gets a saddle point. In Fig. 1.17(b) the two fixed points $u_0 = \pm r_0$ change their roles.



(a) (b)

Fig. 1.16 Phase portrait. The values are: **(a)** $\hat{g}_{jk}^{ms}$ = 0.25 , $\hat{g}_{jk}^{ms}$ = 0.25; **(b)** $\hat{g}_{jk}^{ms}$ = - 0.25. In both cases c_{jk}^{ms} = 1.5 and d_{jk}^{ms}=- 0.3.

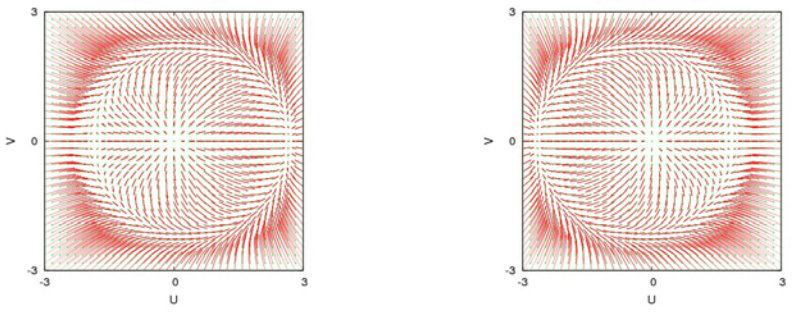


Fig. 1.17 Phase portrait. The values are: **(a)** $\hat{g}_{jk}^{ms}$= 0.25 , $\hat{g}_{jk}^{ms}$ = 0.25; **(b)** $\hat{g}_{jk}^{ms}$ = - 0.25. In both cases c_{jk}^{ms} = 1.5 and d_{jk}^{ms} = 0.3.

The interpretation of e.g. Fig. 1.17(a) in agent view can be given as follows. There are three agents ,we call them organisators, the organisator in the origin (repulsive fixed point) sends all messages (bosonic fields) to its neighbour to the left (saddle point) and to the right (attractive fixed point). The organisator to the right collects all messages that flow in asymptotically to the circle to this agent. Only the messages that start in direction of the circle from the organisator to the left stay on the circle until they reach the right organisator. All messages that are generated by other agents that are outside of the circle also reach the right organisator. All messages that are generated by the organisator in the origin and that are coming from outside of the circle build together an asymptotic flow that can be considered as a fibration (foliation) where the circle is considered as an unstable manifold W^u that defines a base space of a bundle.

1.2.4 Coupling of the Disturbed Eigenanteil Equations

Here our approach is oriented on equations for multimode laser (Haken, 1985). We do this step by step and start with a small perturbation of the coupled Eigenanteile. This means that we just add particle number operators for each type of quantized bosonic fields e.g. $\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}$ that counts the number of "**ms**-messages" (n_{ms}). This approach reads as:

$$\begin{aligned}\frac{d}{dt}\mathbf{ms}_{jk}^{\dagger} = {} & c_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + c_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} - d_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \\ & (c_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger} - d_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk})) + \hat{g}_{jk}^{ms}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + \hat{g}_{jk}^{cr}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \\ & \hat{g}_{jk}^{mt}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}). \end{aligned} \tag{1.60}$$

We set (with indices dropped) $\mathbf{ms} = r_1 e^{i\varphi_1}$, $\mathbf{cr} = r_2 e^{i\varphi_2}$, $\mathbf{mt}^{\dagger} = r_3 e^{i\varphi_3}$, and get the transformed equation:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{r}_1 - \mathrm{i}\mathrm{r}_1\frac{\mathrm{d}}{\mathrm{dt}}\varphi_1 = \mathrm{c}^{\mathrm{ms}}\mathrm{r}_1 - \mathrm{d}^{\mathrm{ms}}(\mathrm{r}_1)^3 + \mathrm{c}^{\mathrm{cr}}\mathrm{r}_2\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_2)} - \mathrm{d}^{\mathrm{cr}}(\mathrm{r}_2)^3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_2)} + \\ \mathrm{c}^{\mathrm{mt}}\mathrm{r}_3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_3)} - \mathrm{d}^{\mathrm{mt}}(\mathrm{r}_3)^3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_3)} + (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\mathrm{e}^{\mathrm{i}\varphi_1}. \tag{1.61}$$

Here we can set $(\mathrm{r}_1)^2 = \mathrm{n}_{\mathrm{ms}}$, etc. After the splitting of this formula into a real part and imaginary part, the resulting equations are:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{r}_1 = \mathrm{c}^{\mathrm{ms}}\mathrm{r}_1 - \mathrm{d}^{\mathrm{ms}}(\mathrm{r}_1)^3 + (\mathrm{c}^{\mathrm{cr}}\mathrm{r}_2 - \mathrm{d}^{\mathrm{cr}}(\mathrm{r}_2)^3)\cos(\varphi_1-\varphi_2) + \\ (\mathrm{c}^{\mathrm{mt}}\mathrm{r}_3 - \mathrm{d}^{\mathrm{mt}}(\mathrm{r}_3)^3)\cos(\varphi_1-\varphi_3) + (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\cos(\varphi_1). \tag{1.62}$$

$$\mathrm{r}_1\frac{\mathrm{d}}{\mathrm{dt}}\varphi_1 = (\mathrm{c}^{\mathrm{cr}}\mathrm{r}_2 - \mathrm{d}^{\mathrm{cr}}(\mathrm{r}_2)^3)\sin(\varphi_1-\varphi_2) + (\mathrm{c}^{\mathrm{mt}}\mathrm{r}_3 - \mathrm{d}^{\mathrm{mt}}(\mathrm{r}_3)^3) \\ \sin(\varphi_1-\varphi_3) + (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\sin(\varphi_1). \tag{1.63}$$

We get equivalent formulas for the two other creation operators (the annihilation operators deliver identical equations as the creation operators).

In the next step we multiply the summed and weighted particle number expression $\hat{\mathrm{g}}^{\mathrm{ms}}_{\mathrm{jk}}(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{cr}}_{\mathrm{jk}}(\mathbf{cr}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{mt}}_{\mathrm{jk}}(\mathbf{mt}^{\dagger}_{\mathrm{jk}}\mathbf{mt}_{\mathrm{jk}})$ with the different creation operators e.g. with $\mathbf{ms}^{\dagger}_{\mathrm{jk}}$. This addition models the fact that the inversion (more fermionic message receiver are in a higher state (excited) than in a lower basic states; named saturation in laser technology) will be reduced by all three fields (modes):

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}^{\dagger}_{\mathrm{jk}} = \mathrm{c}^{\mathrm{ms}}_{\mathrm{jk}}\mathbf{ms}^{\dagger}_{\mathrm{jk}} - \mathrm{d}^{\mathrm{ms}}_{\mathrm{jk}}\mathbf{ms}^{\dagger}_{\mathrm{jk}}(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}) + \mathrm{c}^{\mathrm{cr}}_{\mathrm{jk}}\mathbf{cr}^{\dagger}_{\mathrm{jk}} - \mathrm{d}^{\mathrm{cr}}_{\mathrm{jk}}\mathbf{cr}^{\dagger}_{\mathrm{jk}}(\mathbf{cr}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}) + \\ \mathrm{c}^{\mathrm{mt}}_{\mathrm{jk}}\mathbf{mt}^{\dagger}_{\mathrm{jk}} - \mathrm{d}^{\mathrm{mt}}_{\mathrm{jk}}\mathbf{mt}^{\dagger}_{\mathrm{jk}}(\mathbf{mt}^{\dagger}_{\mathrm{jk}}\mathbf{mt}_{\mathrm{jk}}) + \\ \Big(\hat{\mathrm{g}}^{\mathrm{ms}}_{\mathrm{jk}}(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{cr}}_{\mathrm{jk}}(\mathbf{cr}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{mt}}_{\mathrm{jk}}(\mathbf{mt}^{\dagger}_{\mathrm{jk}}\mathbf{mt}_{\mathrm{jk}})\Big)\mathbf{ms}^{\dagger}_{\mathrm{jk}} + \\ \Big(\hat{\mathrm{g}}^{\mathrm{ms}}_{\mathrm{jk}}(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{cr}}_{\mathrm{jk}}(\mathbf{cr}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{mt}}_{\mathrm{jk}}(\mathbf{mt}^{\dagger}_{\mathrm{jk}}\mathbf{mt}_{\mathrm{jk}})\Big)\mathbf{cr}^{\dagger}_{\mathrm{jk}} + \\ \Big(\hat{\mathrm{g}}^{\mathrm{ms}}_{\mathrm{jk}}(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{cr}}_{\mathrm{jk}}(\mathbf{cr}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}) + \hat{\mathrm{g}}^{\mathrm{mt}}_{\mathrm{jk}}(\mathbf{mt}^{\dagger}_{\mathrm{jk}}\mathbf{mt}_{\mathrm{jk}})\Big)\mathbf{mt}^{\dagger}_{\mathrm{jk}}. \tag{1.64}$$

The solution of this equation reads now:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{r}_1 - \mathrm{i}\mathrm{r}_1\frac{\mathrm{d}}{\mathrm{dt}}\varphi_1 = \mathrm{c}^{\mathrm{ms}}\mathrm{r}_1 - \mathrm{d}^{\mathrm{ms}}(\mathrm{r}_1)^3 + \mathrm{c}^{\mathrm{cr}}\mathrm{r}_2\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_2)} - \mathrm{d}^{\mathrm{cr}}(\mathrm{r}_2)^3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_2)} + \\ \mathrm{c}^{\mathrm{mt}}\mathrm{r}_3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_3)} - \mathrm{d}^{\mathrm{mt}}(\mathrm{r}_3)^3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_3)} + \\ (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\mathrm{r}_1\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_1)} + \\ (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\mathrm{r}_2\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_2)} + \\ (\hat{\mathrm{g}}^{\mathrm{ms}}(\mathrm{r}_1)^2 + \hat{\mathrm{g}}^{\mathrm{cr}}(\mathrm{r}_2)^2 + \hat{\mathrm{g}}^{\mathrm{mt}}(\mathrm{r}_3)^2)\mathrm{r}_3\mathrm{e}^{\mathrm{i}(\varphi_1-\varphi_3)}. \tag{1.65}$$

Finally we include cubic operator terms like $\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{cr}_{\mathrm{jk}}$, they describe the indirect interactions between the bosonic fields (field modes) that are initiated by their interactions with the fermionic agents $\mathbf{a}^{\dagger}_{\mathrm{j}}$ and $\mathbf{a}_{\mathrm{j}}$. For reason of simplicity we do not

write down new coupling coefficients (for precise calculations they must be adapted and changed)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{jk}^{\dagger} =& c_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + c_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} - d_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \\ & c_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger} - d_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}) + \\ & \left(\hat{g}_{jk}^{ms}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + \hat{g}_{jk}^{cr}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \hat{g}_{jk}^{mt}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk})\right)\mathbf{ms}_{jk}^{\dagger} + \\ & \left(\hat{g}_{jk}^{ms}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + \hat{g}_{jk}^{cr}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \hat{g}_{jk}^{mt}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk})\right)\mathbf{cr}_{jk}^{\dagger} + \\ & \left(\hat{g}_{jk}^{ms}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + \hat{g}_{jk}^{cr}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \hat{g}_{jk}^{mt}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk})\right)\mathbf{mt}_{jk}^{\dagger} + \\ & \hat{g}_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}^{\dagger}\mathbf{cr}_{jk} + \hat{g}_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}^{\dagger}\mathbf{mt}_{jk} + \hat{g}_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}^{\dagger}\mathbf{ms}_{jk} + \\ & \hat{g}_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}^{\dagger}\mathbf{mt}_{jk} + \hat{g}_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}^{\dagger}\mathbf{ms}_{jk} + \hat{g}_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}^{\dagger}\mathbf{cr}_{jk}.\end{aligned} \tag{1.66}$$

The result is:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}r_1 - ir_1\frac{\mathrm{d}}{\mathrm{dt}}\varphi_1 =& c^{ms}r_1 - d^{ms}(r_1)^3 + c^{cr}r_2e^{i(\varphi_1-\varphi_2)} - d^{cr}(r_2)^3e^{i(\varphi_1-\varphi_2)} + \\ & c^{mt}r_3e^{i(\varphi_1-\varphi_3)} - d^{mt}(r_3)^3e^{i(\varphi_1-\varphi_3)} + (\hat{g}^{ms}(r_1)^2 + \\ & \hat{g}^{cr}(r_2)^2 + \hat{g}^{mt}(r_3)^2)r_1e^{i(\varphi_1-\varphi_1)} + \\ & (\hat{g}^{ms}(r_1)^2 + \hat{g}^{cr}(r_2)^2 + \hat{g}^{mt}(r_3)^2)r_2e^{i(\varphi_1-\varphi_2)} + \\ & (\hat{g}^{ms}(r_1)^2 + \hat{g}^{cr}(r_2)^2 + \hat{g}^{mt}(r_3)^2)r_3e^{i(\varphi_1-\varphi_3)} + \\ & \hat{g}_{jk}^{ms}(r_1)^2r_2e^{i(\varphi_2-\varphi_1)} + \hat{g}_{jk}^{ms}(r_1)^2r_3e^{i(\varphi_3-\varphi_1)} + \\ & \hat{g}_{jk}^{cr}(r_2)^2r_1e^{i2(\varphi_1-\varphi_2)} + \hat{g}_{jk}^{cr}(r_2)^2r_3e^{i(\varphi_1+\varphi_3-2\varphi_2)} + \\ & \hat{g}_{jk}^{mt}(r_3)^2r_1e^{i2(\varphi_1-\varphi_3)} + \hat{g}_{jk}^{mt}(r_3)^2r_2e^{i(\varphi_1+\varphi_2-2\varphi_3)}.\end{aligned} \tag{1.67}$$

One additional possible quadratic direct coupling of the bosonic operators that is initiated by similar atomic interactions (Zaslavsky, 2007) is given by the following equations:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{jk}^{\dagger} =& c_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger} - d_{jk}^{ms}\mathbf{ms}_{jk}^{\dagger}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + c_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} - d_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \\ & c_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger} - d_{jk}^{mt}\mathbf{mt}_{jk}^{\dagger}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}) + \\ & g_{jk}\Big((\mathbf{ms}_{jk}^{\dagger})^2 - (\mathbf{ms}_{jk})^2 + (\mathbf{cr}_{jk}^{\dagger})^2 - (\mathbf{cr}_{jk})^2 + (\mathbf{mt}_{jk}^{\dagger})^2 - (\mathbf{mt}_{jk})^2\Big).\end{aligned} \tag{1.68}$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{ms}_{jk} =& c_{jk}^{ms}\mathbf{ms}_{jk} - d_{jk}^{ms}\mathbf{ms}_{jk}(\mathbf{ms}_{jk}^{\dagger}\mathbf{ms}_{jk}) + c_{jk}^{cr}\mathbf{cr}_{jk}^{\dagger} - d_{jk}^{cr}\mathbf{cr}_{jk}(\mathbf{cr}_{jk}^{\dagger}\mathbf{cr}_{jk}) + \\ & c_{jk}^{mt}\mathbf{mt}_{jk} - d_{jk}^{mt}\mathbf{mt}_{jk}(\mathbf{mt}_{jk}^{\dagger}\mathbf{mt}_{jk}) + \\ & g_{jk}\Big((\mathbf{ms}_{jk}^{\dagger})^2 - (\mathbf{ms}_{jk})^2 - (\mathbf{cr}_{jk}^{\dagger})^2 + (\mathbf{cr}_{jk})^2 + (\mathbf{mt}_{jk})^2 - (\mathbf{mt}_{jk}^{\dagger})^2\Big).\end{aligned} \tag{1.69}$$

There is not enough space in this subchapter to present all the details of the graphical solutions of all before mentioned distorted equations, but we have observed in parts that a tendency to asymptotic stability of the QFT-approach can be observed, if the coupling constants, the damping constants κ, and γ, and the relaxation time T are not space-time or even field dependent. The coupled equations "prefer" to decouple in direction to their Eigenanteile. A proof of this proposition can not be given at this time since this a problem in a very high parameter space and we are still on the way to perform in a systematic manner the necessary calculations, and we know that also chaotic solutions exist.

1.2.5 Information Model and Interactions of Structured Components

1.2.5.1 Interaction Revisited

The interaction of fermionic agents and bosonic message ("force") fields that represent different signal quanta like photons, intracellular signaling proteins or extracellular signaling proteins (e.g. synaptic, endocrine (hormone based), etc.) will be modeled in the first approach again by the interaction Hamilton operator (1.5):

$$\begin{aligned}\mathbf{H}_{\mathrm{I}} = \mathrm{i}\hbar \sum_{\mathrm{j,k,l,m,n}} \mathrm{g}^{\mathrm{ms}}_{\mathrm{jk}} \left(\boldsymbol{\alpha}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\mathbf{ms}^{\dagger}_{\mathrm{jk}}(\mathrm{n}) - \boldsymbol{\alpha}^{\dagger}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\mathbf{ms}_{\mathrm{jk}}(\mathrm{n}) \right) + \\ \mathrm{i}\hbar \sum_{\mathrm{j,k,l,m,n}} \mathrm{g}^{\mathrm{cr}}_{\mathrm{jk}} \left(\boldsymbol{\alpha}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\mathbf{cr}^{\dagger}_{\mathrm{jk}}(\mathrm{n}) - \boldsymbol{\alpha}^{\dagger}_{\mathrm{j}}(\mathrm{m},\ \mathrm{l})\mathbf{cr}_{\mathrm{jk}}(\mathrm{n}) \right) + \\ \mathrm{i}\hbar \sum_{\mathrm{j,k,l,m,n}} \mathrm{g}^{\mathrm{mt}}_{\mathrm{jk}} \left(\boldsymbol{\alpha}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\mathbf{mt}^{\dagger}_{\mathrm{jk}}(\mathrm{n}) - \boldsymbol{\alpha}^{\dagger}_{\mathrm{j}}(\mathrm{m},\mathrm{l})\mathbf{mt}_{\mathrm{jk}}(\mathrm{n}) \right).\end{aligned} \tag{1.70}$$

We use the state flip (transition) operators $\boldsymbol{\alpha}^{\dagger}_{\mathrm{j}}$ and $\boldsymbol{\alpha}_{\mathrm{j}}$, that have been introduced in Sect. 1.2.1; further we consider the state l (e.g. $l = 0$, here we use the cursive l in order to avoid confusion with the number 1) as resting state, and we use the earlier mode index k as an identification of fermionic agent. The absorption of a message brings the receiver into an excited state $\mathrm{m} = 1$. We repeated above the previous defined interaction Hamiltonian since the application of the new definition of information (calculated by operators that create or annihilate quantized fields) implicates a modified version of $\mathbf{H}_{\mathrm{I}}$ that is primarily based on mutual message exchange. Being on the way to present the QFT-based description of information as an operator whose expectation values depend on the involved states (can be considered a context awareness) we have to consider shortly two possible probability distributions of the message exchange since the kind of message exchange influences the definition of information.

In addition we assume the coherence of all message exchanges (above the laser threshold that is considered as a control parameter that defines a positive net gain) so all expectation values of these messages obey the Poisson distribution. Hereby all the different messages (of the fixed type n) that are exchanged in the activity

phase of the operator pair can be considered as a "message-field" that responds to a certain distribution function. In our case of coherency this is the Poisson distribution for the number N of transmitted messages of type n (photonic statistics). The kind of statistics can be expressed by the number operator e.g. for $\mathbf{N_{ms}} = \mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}$. The variance (dropping of the indices **ms**) of this message number is defined by

$$\langle(\mathbf{N} - \langle\mathbf{N}\rangle)^2\rangle = \langle\mathbf{N}^2\rangle - \langle\mathbf{N}\rangle^2 = \langle(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}})^2\rangle - \langle\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}\rangle^2.$$

For the Poisson distribution function the following approximation is true $\langle\mathbf{N}^2\rangle - \langle\mathbf{N}\rangle^2 \approx \langle\mathbf{N}\rangle$. This means that the messages are send in a fixed averaged time distance. If the Bose-Einstein statistics is fulfilled then the value of the variance is different $\langle\mathbf{N}^2\rangle - \langle\mathbf{N}\rangle^2 = \langle(\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}})^2\rangle - \langle\mathbf{ms}^{\dagger}_{\mathrm{jk}}\mathbf{ms}_{\mathrm{jk}}\rangle^2 = \langle\mathbf{N}\rangle(\langle\mathbf{N}\rangle + 1)$. The relevance of this distribution function is the lumping of messages, the ramification is the decoherence of the field and therefore no cooperation between fermionic agents come about.

1.2.5.2 Postulation of Information Rules

We define the concept of information by four characteristic rules:

1. *Synchronisation*. There is a mechanism that synchronises the communication between two or more agents (partners) meaning the semantic compatibility of the agents that initiate a communication session. In practice such a communication channel is handcrafted by a developer.
2. *Compatibility*. The receiver of a message has in the sense of Pulitzer not only to acknowledge the received message; but even more important is the demand that the receiver "understands" the matter of the sender and behaves (reacts) in a manner as it is assumed by the sender (Haken, 1988). In more detail this request denotes that both communication partners are in a configuration where they are compatible in their internal states $\mathrm{S_j}$, in their knowledge $\mathrm{W_{jk}}$ and even more important their distance of information $\mathrm{dist}(\mathrm{Inf_j}, \mathrm{Inf_k})$ is below a given threshold. If all conditions are fulfilled then a semantic equivalence is available.
3. *Component Building*. Two components (agents) j and k continue the "negotiation" concerning their combination as long as their common knowledge $\mathrm{W_{jk}}$ is minimised (similar status of knowledge for both agents defined by a symmetric Kullback measure), the expectation values of the individuals state operators $\mathrm{S_j} = \langle\mathbf{S_j}\rangle = \mathrm{tr}(\boldsymbol{\rho}\mathbf{S_j})$ and $\mathrm{S_k} = \langle\mathbf{S_j}\rangle = \mathrm{tr}(\boldsymbol{\rho}\mathbf{S_k})$ are maximised, and the common information $\mathrm{Inf_j} = \langle\mathbf{Inf_j}\rangle = \mathrm{tr}\langle\boldsymbol{\rho}\mathbf{Inf_j}\rangle$ of agent j is minimised (for more detailed calculations see Sect. 3.3.2).
4. *Open ended evolution*. The process of information collection is iterative and describes three basic algorithms. These are the information that describe the genotype (defining the evolution and fitness), the phenotype (defining the behaviour and higher cognitive abilities) and the controllers coordinating the gathering of information (learning) concerning evolution and fitness and the interactions of these two different learning procedures. The whole process ends if a minimum (infimum) of total information in a given environment has been achieved

(open systems). Here one open question is how the fitness of a given individual influences its genome in that different species emerge that are genetically incompatible.

Here $\boldsymbol{\rho}$ is the density operator (density matrix) and tr is the trace of a matrix: e.g. $\mathrm{tr}(\boldsymbol{\rho}\mathbf{S}_\mathrm{j}) = \sum_\mathrm{m} \langle \mathrm{m} \left| \boldsymbol{\rho}\mathbf{S}_\mathrm{j} \right| \mathrm{m} \rangle$. The total individual state operator $\mathbf{S}_\mathrm{j}$ of an agent j with several internal states (state sum) is defined by:

$$\mathbf{S}_\mathrm{j} = \sum_\mathrm{k} \mathrm{e}^{\left(\boldsymbol{\mu}_\mathrm{j}\left(\langle \mathbf{N}_\mathrm{jk}^2 \rangle - \langle \mathbf{N}_\mathrm{jk} \rangle^2\right) - \mathrm{W}_\mathrm{jk}\right)/\langle \mathbf{N}_\mathrm{j}^2 \rangle}, \tag{1.71}$$

where $\langle \mathbf{N}_\mathrm{jk} \rangle = \langle \mathbf{ms}_\mathrm{jk}^\dagger \mathbf{ms}_\mathrm{jk} \rangle$ is the mean value of messages that agents j and k exchange, and $\langle \mathbf{N}_\mathrm{jk}^2 \rangle = \langle \mathbf{ms}_\mathrm{jk}^\dagger \mathbf{ms}_\mathrm{jk} \mathbf{ms}_\mathrm{jk}^\dagger \mathbf{ms}_\mathrm{jk} \rangle$ is the expectation value of $\mathbf{N}_\mathrm{jk}^2$. The expression $(\langle \mathbf{N}_\mathrm{jk}^2 \rangle - \langle \mathbf{N}_\mathrm{jk} \rangle^2)$ defines the expectation value of the quadratic fluctuations. The knowledge W_jk is defined as the symmetric Kullback measure of the two probability distributions of agents j and k, (see Sect. 3.3, Eq. 3.8).

The activities of an agent can be compared to that one of a chemical potential that characterise the possibilities of a substance to interact with other substrates, to transfer into other states and to distribute all over the space. A reaction, conversion, and redistribution can only occur without enforcement if the potential in the initial state is greater as in the final state. These features are also relevant to statistical physics where we have borrowed our definition (here we neglect for brevity other contributions e.g. of $\mathbf{cr}_\mathrm{jk}, \mathbf{mt}_\mathrm{jk}$)

$$\boldsymbol{\mu}_\mathrm{j} = \sum_{\mathrm{k}=0}^{\mathrm{K}} \sum_{\mathrm{m_j}=0}^{\mathrm{M}} \ln\left(\mathrm{g}_\mathrm{jk}^\mathrm{ms} \mathbf{a}_\mathrm{j}^\dagger(\mathrm{m_j}) \mathbf{ms}_\mathrm{jk} + \mathrm{g}_\mathrm{jk}^{\mathrm{ms}*}\left(\mathbf{a}_\mathrm{j}(\mathrm{m_j}) \mathbf{ms}_\mathrm{jk}^\dagger\right)\right). \tag{1.72}$$

The quantisation of $\boldsymbol{\mu}_\mathrm{j}$ is performed by the inclusion of quantum field operators and hereby finally we get the quantisation of the information. Further benefits of the approach with the "chemical potential" are the possibility to combine different agents to a bigger unit, to model diffusion and the possibility to describe the adaptation of an agent performed by different phase transitions (e.g. solid state, fluid state, gaseous state). This can be initiated by special messages and appropriate coupling constants that generate an appropriate state transition. Thus it is possible to consider e.g. environmental restrictions as dedicated messages from outside.

The operator of the "statistical potential" of an agent j, after it evaluates all messages it has received, is defined by:

$$\boldsymbol{\Omega}_\mathrm{j} = -\langle \mathbf{N}_\mathrm{j}^2 \rangle \ln\left(\sum_\mathrm{k} e^{\left(\mu_\mathrm{j}\left(\langle \mathbf{N}_\mathrm{jk}^2 \rangle - \langle \mathbf{N}_\mathrm{jk} \rangle^2\right) - \mathrm{W}_\mathrm{jk}\right)/\langle \mathbf{N}_\mathrm{j}^2 \rangle}\right) = -\langle \mathbf{N}_\mathrm{j}^2 \rangle \ln \mathbf{S}_\mathrm{j}. \tag{1.73}$$

Since we consider $\boldsymbol{\mu}_\mathrm{j}$ as a dynamic variable the state operator $\mathbf{S}_\mathrm{j}$ can be characterised beside physical or chemical terms also by states of nonlinear dynamics like equilibrium states, periodic states or even chaotic states. Finally we define the operator of information as the derivative of the operational "statistical potential" with

respect to $\left\langle \mathbf{N}_j^2 \right\rangle$ that correspondsd to the temperature of the state S_j in statistical physics:

$$\mathbf{Inf}_j = -\frac{\partial \boldsymbol{\Omega}_j}{\partial \left\langle \mathbf{N}_j^2 \right\rangle}. \tag{1.74}$$

Transferred to our approach this means that the synchronisation of two agents (e.g. robot-cells) is guaranteed, since the communicative coherence and message exchange symmetry between all agents is established. Expressed in a new interaction Hamiltonian this might look like:

$$\begin{aligned}\mathbf{H}_I' =& i\hbar \sum_{j,k} g_{jk}^{ms} \mathbf{ms}_{jk}^{\dagger} \mathbf{ms}_{jk} \left(\boldsymbol{\alpha}_j(m_j,0)\boldsymbol{\alpha}_k^{\dagger}(m_k,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0)\boldsymbol{\alpha}_k(m_k,0) \right) + \\ & i\hbar \sum_{j,k} g_{jk}^{cr} \mathbf{cr}_{jk}^{\dagger} \mathbf{cr}_{jk} \left(\boldsymbol{\alpha}_j(m_j,0)\boldsymbol{\alpha}_k^{\dagger}(m_k,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0)\boldsymbol{\alpha}_k(m_k,0) \right) + \\ & i\hbar \sum_{j,k} g_{jk}^{mt} \mathbf{mt}_{jk}^{\dagger} \mathbf{mt}_{jk} \left(\boldsymbol{\alpha}_j(m_j,0)\boldsymbol{\alpha}_k^{\dagger}(m_k,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0)\boldsymbol{\alpha}_k(m_k,0) \right).\end{aligned} \tag{1.75}$$

This interaction Hamiltonian fulfills the rule of synchronisation mentioned before (rule 1). The equation of motion for $\boldsymbol{\alpha}_j^{\dagger}(m_j,0)$ are:

$$\begin{aligned}\frac{d}{dt}\boldsymbol{\alpha}_j^{\dagger}(m_j,0) =& \frac{i}{\hbar}[\mathbf{H}_I', \boldsymbol{\alpha}_j^{\dagger}(m_j,0)] \\ =& 2\sum_k g_{jk}^{ms} \mathbf{ms}_{jk}^{\dagger} \mathbf{ms}_{jk} \left(\boldsymbol{\alpha}_j^{\dagger}(0,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0) \right) + \\ & 2\sum_k g_{jk}^{cr} \mathbf{cr}_{jk}^{\dagger} \mathbf{cr}_{jk} \left(\boldsymbol{\alpha}_j^{\dagger}(0,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0) \right) + \\ & 2\sum_k g_{jk}^{mt} \mathbf{mt}_{jk}^{\dagger} \mathbf{mt}_{jk} \left(\boldsymbol{\alpha}_j^{\dagger}(0,0) - \boldsymbol{\alpha}_j^{\dagger}(m_j,0) \right).\end{aligned} \tag{1.76}$$

Consequently we can solve the equations of motion for all operators that are part of the definition of the chemical potential $\boldsymbol{\mu}_j$, then with the aid of the knowledge W_{jk} and the aid of the "statistical potential" $\boldsymbol{\Omega}_j$ a value for Inf_j can be calculated. If the knowledge of two robot-cells j and k are compatible (e.g. the difference between Inf_j and Inf_k is below a threshold) then not only the dynamic of the information flow between every two agents but also the total information flow can be calculated and analysed (e.g. by phase portraits) whether the combination of two robot-cells or even of an organisms is a stable equilibrium state or this state is unstable.

1.2.5.3 Structured Objects

The activities of different genes/cells are usually clearly different and they might have strong coupling constants. This fact implies that such units can operate as "seeds" for agent-based (morphological) networks that are constructed by strong

message exchange and can be considered as the predefinition of a final structure. The result is a topological framework that describes the structure of an organism.

We try to demonstrate our approach by the example of water. A single water molecule is build up by two hydrogen atoms H_1, H_2 and one molecule O that are bounded together by two covalent bonds. This means that there is a communication only between H_1 and O, respectively H_2 and O. A possible interaction Hamiltonian is, where we only mention **ms** and $\mathbf{ms}^\dagger$ that represent primarily external effects (dominant internal processes are manifested in a Hamiltonian that owns the same structure):

$$\begin{aligned}
\mathbf{H}_{\mathrm{I}}^{\mathrm{H_2O}} = \mathrm{i}\hbar \mathrm{g}_{\mathrm{HO}}^{\mathrm{ms}} \Big(& \left(\mathbf{ms}_{\mathrm{H_1O}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{H_1}} (\mathrm{m}_{\mathrm{H_1}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_1O}} \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{O1}}, 0)\right) \\
& \left(\mathbf{ms}_{\mathrm{H_2O}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{H_2}} (\mathrm{m}_{\mathrm{H_2}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_2O}} \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) - \\
& \left(\mathbf{ms}_{\mathrm{H_2O}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_2O}} \boldsymbol{\alpha}_{\mathrm{H_2}}^{\dagger} (\mathrm{m}_{\mathrm{H_{2,0}}})\right) \\
& \left(\mathbf{ms}_{\mathrm{H_1O}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_1}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_1O}} \boldsymbol{\alpha}_{\mathrm{H_1}}^{\dagger} (\mathrm{m}_{\mathrm{H_1}}, 0)\right) \Big) \\
= \mathrm{i}\hbar \mathrm{g}_{\mathrm{HO}}^{\mathrm{ms}} \Big(& \left(\mathbf{ms}_{\mathrm{H_1O}}^{\dagger} \mathbf{ms}_{\mathrm{H_1O}} \mathbf{ms}_{\mathrm{H_2O}}^{\dagger} \mathbf{ms}_{\mathrm{H_2O}}\right) \\
& \left(\boldsymbol{\alpha}_{\mathrm{H_1}} (\mathrm{m}_{\mathrm{H_1}}, 0) \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{O1}}, 0) \boldsymbol{\alpha} \mathrm{H}_2 (\mathrm{m}_{\mathrm{H_{2,0}}}) \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) - \\
& \left(\mathbf{ms}_{\mathrm{H_2O}}^{\dagger} \mathbf{ms}_{\mathrm{H_2O}} \mathbf{ms}_{\mathrm{H_1O}}^{\dagger} \mathbf{ms}_{\mathrm{H_1O}}\right) \\
& \left(\boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_2}}, 0) \boldsymbol{\alpha}_{\mathrm{H_2}}^{\dagger} (\mathrm{m}_{\mathrm{H_2}}, 0) \boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_1}}^{\dagger} (\mathrm{m}_{1}, 0)\right) \Big) \\
= \mathrm{i}\hbar \mathrm{g}_{\mathrm{HO}}^{\mathrm{ms}} \Big(& \mathbf{N}_{\mathrm{H_1O}}^{\mathrm{ms}} \mathbf{N}_{\mathrm{H_2O}}^{\mathrm{ms}} \left(\boldsymbol{\alpha}_{\mathrm{H_1}} (\mathrm{m}_{\mathrm{H_1}}, 0) \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{0_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_2}} (\mathrm{m}_{\mathrm{H_2}}, 0) \boldsymbol{\alpha}_{\mathrm{O}}^{\dagger} (\mathrm{m}_{\mathrm{0_2}}, 0)\right) - \\
& \mathbf{N}_{\mathrm{H_2O}}^{\mathrm{ms}} \mathbf{N}_{\mathrm{H_1O}}^{\mathrm{ms}} \left(\boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_2}}, 0) \boldsymbol{\alpha}_{\mathrm{H_2}}^{\dagger} (\mathrm{m}_{\mathrm{H_2}}, 0) \boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_1}}^{\dagger} (\mathrm{m}_{\mathrm{H_1}}, 0)\right) \Big)
\end{aligned} \tag{1.77}$$

Here $\mathbf{N}_{\mathrm{H_1O}}^{\mathrm{ms}}$ and $\mathbf{N}_{\mathrm{H_2O}}^{\mathrm{ms}}$ are the "message counter operators" and the index H_2O does not mean the water molecule but the message exchange between the second hydrogen atom and the oxygen. This approach can be considered as generalised information based diffusion approach in the sense that between all participants there is a two-way synchronised message exchange. But till now the connection of this Hamiltonian with the standard theory e.g. (Haken & Wolf, 1998) is not obvious, and indeed, the consensus with the reality is still open and still has to be proofed.

The next question that arises is how we can model the structure of a H_2O molecule. This can only be done if we not only consider the message exchange but also the activities that the receivers perform in consequence of exchange of "news". In our case we must consider the subelements of the three molecules. These are the electrons (more precise: the interactions of the orbitals of the participating

electrons) that make the covalent bond (overlapping of adjacent atomic orbitals) polar. In more detail we must consider the interaction of the two $1s^1$ orbitals of the hydrogen atoms with the $2p^4$ orbit of the oxygen under the strong observation of the Pauli Principe. Here we do not delve into the details of quantum physic (respectively physical chemistry) but argue with the relevant symmetry groups for the H_2O molecule that consider all orbitals together. The drawback of this procedural message is that this kind of the group theory cannot explain why the water molecule is stable but e.g. He_2 is unstable. Such stability declaration can usually only be given by direct calculations of the interaction patterns.

The oxygen nucleus draws electron orbits away and as a result an electrical tetraeder structure is generated (Alberts *et al.*, 2008); the symmetry group is therefore T_d (full notation: $\overline{4}3m$). The spatial structure of it is defined by C_{2v}.

The stabilities of individual parts can again be calculated e.g. by

$$\frac{\mathrm{d}}{\mathrm{dt}}(\mathbf{ms}^{\dagger}_{\mathrm{H_1O}}\boldsymbol{\alpha}_{\mathrm{H_1}}(\mathrm{m_{H_1}},0)) = \frac{\mathrm{i}}{\hbar}\left[\mathbf{H}_{\mathrm{I}}^{\mathrm{H_2O}}, (\mathbf{ms}^{\dagger}_{\mathrm{H_1O}}\boldsymbol{\alpha}_{\mathrm{H_1}}(\mathrm{m_{H_1}},0))\right] \tag{1.78}$$

and analysed by the methods of nonlinear dynamics as we have done before.

The energy calculation can be done by the standard model of harmonic oscillator. Here we couple two oscillators together. The non interacting Hamiltonian is just the well known sum of the two separate oscillators

$$\mathbf{H} = \hbar\sum_{\underline{\mathrm{i}}=1}^{2}\omega_{\underline{\mathrm{p}}_{\underline{\mathrm{i}}}}\left(\mathbf{a}^{\dagger}_{\underline{\mathrm{p}}_{\underline{\mathrm{i}}}}\mathbf{a}_{\underline{\mathrm{p}}_{\underline{\mathrm{i}}}} + \frac{1}{2}\right), \tag{1.79}$$

where $\underline{\mathrm{p}}_{\underline{\mathrm{i}}}$ is a 3-d momentum, and the eigenfunctions are also well known (Hermite polynomials).

The reader recognised certainly that the symmetry groups of the H_2O molecule are not really relevant to the connection of some few robot-cells to a structured component of a symbiotic organism, but we operate here again in strong analogy to this algebraic method. The interaction between the participating active cells is modeled by the contents of the exchanged messages. Thus we can describe whether two cells own similar genomes and therefore are compatible to aggregate to a major component. A message can also characterise e.g. pure physical effects that two cells merge together since they are by chance (probability) sticked together close enough and mutual attractive forces merge them to a new unit. By this approach defects of individual cells can be modeled and detected. The remedy of such defects can be performed by the calculation and comparison of the pairwise exchanged information and thus even modify or exchange malfunctioning robot cells. The Hamiltonian does not deliver these information but it will be used to describe and to analyse the dynamics of the flow of the dedicated messages, and the symmetry of such a flow can be defined by a relevant group. Here we can read in more detail e.g. whether cells are attractive, repulsive, or neutral. Further such a group could deliver a hint which symmetry group governs the information space. In sequence, this knowledge delivers the definition of a metric in such a space and therefore provide us with the correct calculation of the distance of two information values.

Now we turn to the coupling of two water molecules. The two water molecules are connected by a so called hydrogen bond between the oxygen atoms via one mediating hydrogen molecule. We model this by the following Hamiltonian:

$$\begin{aligned}
\mathbf{H}_{\mathrm{I}}^{\mathrm{water}} = \mathrm{i}\hbar \mathrm{g}_{\mathrm{OO}}^{\mathrm{ms}} \Big(& \left(\mathbf{ms}_{\mathrm{O_1H_3}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{O_1}} (\mathrm{m}_{\mathrm{O_1}}, 0)\right) \left(\mathbf{ms}_{\mathrm{O_1H_3}} \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_3}}, 0)\right) \\
& \left(\mathbf{ms}_{\mathrm{H_3O_2}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{H_3}} (\mathrm{m}_{\mathrm{H_3}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_3O_2}} \boldsymbol{\alpha}_{\mathrm{O_2}}^{\dagger} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) - \\
& \left(\mathbf{ms}_{\mathrm{H_3O_2}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{O_2}} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) \left(\mathbf{ms}_{\mathrm{H_3O_2}} \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_{3,0}}})\right) \\
& \left(\mathbf{ms}_{\mathrm{O_1H_3}}^{\dagger} \boldsymbol{\alpha}_{\mathrm{H_3}} (\mathrm{m}_{\mathrm{H_3}}, 0)\right) \left(\mathbf{ms}_{\mathrm{O_1H_3}} \boldsymbol{\alpha}_{\mathrm{O_1}}^{\dagger} (\mathrm{m}_{\mathrm{O_1}}, 0)\right) \Big) \\
= \mathrm{i}\hbar \mathrm{g}_{\mathrm{HO}}^{\mathrm{ms}} \Big(& \left(\mathbf{ms}_{\mathrm{O_1H_3}}^{\dagger} \mathbf{ms}_{\mathrm{O_1H_3}} \mathbf{ms}_{\mathrm{H_3O_2}}^{\dagger} \mathbf{ms}_{\mathrm{H_3O_2}}\right) \\
& \left(\boldsymbol{\alpha}_{\mathrm{O}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_3}}, 0) \boldsymbol{\alpha} \mathrm{H3} \left(\mathrm{m}_{\mathrm{H_{3,0}}}\right) \boldsymbol{\alpha}_{\mathrm{O_2}}^{\dagger} (\mathrm{m}_{\mathrm{O_2}}, 0)\right) - \\
& \left(\mathbf{ms}_{\mathrm{H_3O_2}}^{\dagger} \mathbf{ms}_{\mathrm{H_3O_2}} \mathbf{ms}_{\mathrm{O_1H_3}}^{\dagger} \mathbf{ms}_{\mathrm{O_1H_3}}\right) \\
& \left(\boldsymbol{\alpha}_{\mathrm{O_2}} (\mathrm{m}_{\mathrm{O_2}}, 0) \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_3}}, 0) \boldsymbol{\alpha}_{\mathrm{O_1}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_1}}^{\dagger} (\mathrm{m}_{\mathrm{H_1}}, 0)\right) \Big) \\
= \mathrm{i}\hbar \mathrm{g}_{\mathrm{HO}}^{\mathrm{ms}} \Big(& \mathbf{N}_{O_1H_3}^{ms} \mathbf{N}_{H_3O_2}^{ms} \left(\boldsymbol{\alpha}_{\mathrm{O_1}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_3}}, 0) \boldsymbol{\alpha}_{\mathrm{H_2}} (\mathrm{m}_{\mathrm{H_2}}, 0) \boldsymbol{\alpha}_{\mathrm{O_2}}^{\dagger} (\mathrm{m}_{\mathrm{0_2}}, 0)\right) - \\
& \mathbf{N}_{\mathrm{H_3O_2}}^{\mathrm{ms}} \mathbf{N}_{\mathrm{O_1H_3}}^{\mathrm{ms}} \left(\boldsymbol{\alpha}_{\mathrm{O_2}} (\mathrm{m}_{\mathrm{O_2}}, 0) \boldsymbol{\alpha}_{\mathrm{H_3}}^{\dagger} (\mathrm{m}_{\mathrm{H_3}}, 0) \boldsymbol{\alpha}_{\mathrm{O_1}} (\mathrm{m}_{\mathrm{O_1}}, 0) \boldsymbol{\alpha}_{\mathrm{H_1}}^{\dagger} (\mathrm{m}_{\mathrm{H_1}}, 0)\right) \Big).
\end{aligned} \tag{1.80}$$

Analogous to a symmetry group of water we have to search for a symmetry group of a developed organism. This can usually be done on geometrical level (e.g spherical or cylindrical symmetry) or, as we propose, on the level of the information flows of different organising agents that represent different major components (compartments) of an organism. Here we can study the flow of different message types that are send and received among these agents hereby defining the dynamic type of agent (e.g. attractive) and search for the symmetry groups that describe these flows.

1.2.5.4 Action Integral as Fitness Measure

We employ the action integral

$$\mathrm{W}_{12} = \int_{\mathrm{t_1}}^{\mathrm{t_2}} \mathscr{L}\left(\Phi(\mathrm{x}), \partial_{\mu}\Phi(\mathrm{x}), \Psi(\mathrm{x}), \partial_{\mu}\Phi(\mathrm{x})\right) \mathrm{dt} \tag{1.81}$$

to describe and evaluate the propagation of an action. We will use it to define a measure how fit (efficient) a combined structured object in comparison to its individual components is. Here $\mathscr{L}$ is the Lagrangian density operator, Φ is a bosonic

quantized field, and Ψ is a fermionic field that solves adequate Schrödinger equations. The Lagrangian density operator without interaction is:

$$\begin{aligned}\mathscr{L}_S \;=\; i\hbar\boldsymbol{\Phi}^{\dagger}(x)\,\dot{\boldsymbol{\Phi}}(x) \;+\; i\hbar\boldsymbol{\Psi}^{\dagger}(x)\,\dot{\boldsymbol{\Psi}}(x) \;-\frac{\hbar^2}{2m_{\boldsymbol{\Phi}}}\nabla\boldsymbol{\Phi}^{\dagger}(x)\,\nabla\boldsymbol{\Phi}(x) - \\ \Phi^{\dagger}(\mathrm{x})\,\mathrm{V}_{\Phi}(\mathrm{x})\,\Phi(\mathrm{x}) \;-\frac{\hbar^2}{2\mathrm{m}_{\Psi}}\nabla\Psi^{\dagger}(\mathrm{x})\,\nabla\Psi(\mathrm{x}) \;-\; \Psi^{\dagger}(\mathrm{x})\,\mathrm{V}_{\Psi}(\mathrm{x})\,\Psi(\mathrm{x}).\end{aligned} \tag{1.82}$$

In this formula ∇ is the 3-d Nabla operator and the point defines the temporal derivation. The general correlation between the Schrödinger-densities (Index S) densities and the interaction densities $\mathscr{L}_{\text{total}} \;=\; \mathscr{L}_{\text{S}} \;+\; \mathscr{L}_{\text{I}}$ and $\mathscr{H}_{\text{total}} \;=\; \mathscr{H}_{\text{S}} \;+\; \mathscr{H}_{\text{I}}$ is

$$\mathscr{H}_{\text{total}} \;=\; \pi_{\Phi}\dot{\Phi} \;+\; \pi_{\Phi^{\dagger}}\dot{\Phi}^{\dagger} \;+\; \pi_{\Psi}\dot{\Psi} \;+\; \pi_{\Psi^{\dagger}}\dot{\Psi}^{\dagger} \;-\; \mathscr{L}_{\text{total}}, \tag{1.83}$$

where $\pi_{\Phi} \;=\; \partial\mathscr{L}_{\text{total}}/\partial\dot{\Phi}$, etc. are the conjugated fields. For example we got for $\mathscr{H}_{\text{S}}$ the result $\left(\pi_{\Phi} \;=\; \mathrm{i}\hbar\Phi^{\dagger}, \pi_{\Psi} \;=\; \mathrm{i}\hbar\Psi^{\dagger}\right)$, all remaining conjugate fields are zero:

$$\begin{aligned}\mathscr{H}_{\text{S}} \;=& \frac{\hbar^2}{2\mathrm{m}_{\Phi}}\nabla\Phi^{\dagger}(\mathrm{x})\nabla\Phi(\mathrm{x}) + \Phi^{\dagger}(\mathrm{x})\mathrm{V}_{\Phi}(\mathrm{x})\Phi(\mathrm{x}) + \\ & \frac{\hbar^2}{2\mathrm{m}_{\Psi}}\nabla\Psi^{\dagger}(\mathrm{x})\nabla\Psi(\mathrm{x}) + \Psi^{\dagger}(\mathrm{x})\mathrm{V}_{\Psi}(\mathrm{x})\Psi(\mathrm{x}).\end{aligned} \tag{1.84}$$

The field operator Φ obeys an equal time commutation rule, whereas Ψ fulfills an equal time anti-commutation rule. The stability of a system is usually investigated by the form of potential that defines equilibrium states. In our case we have to fix the two potentials $\mathrm{V}_{\Psi}(\mathrm{x})$ and $\mathrm{V}_{\Phi}(\mathrm{x})$.

Typical potentials might be (Zaslavsky, 2007): the kicked oscillator, where a is the distance of the two positions

$$\begin{aligned}\mathrm{V}_{\Psi}(\mathrm{x}) = \frac{1}{2}\alpha\sum_{\mathrm{j}}\left(\underline{\mathrm{x}}_{\mathrm{j}+1} - \underline{\mathrm{x}}_{\mathrm{j}} - \mathrm{a}\right)^2 - \frac{1}{2}\beta\sum_{\mathrm{j}}\left(\underline{\mathrm{x}}_{\mathrm{j}+2} - \underline{\mathrm{x}}_{\mathrm{j}} - 2\mathrm{a}\right)^2 \\ +\gamma\sum_{\mathrm{j}}\mathrm{m}_{\mathrm{j}}\omega_{\mathrm{j}}^2\cos\left(2\pi\mathrm{j}\frac{\mathrm{t}}{\mathrm{T}}\right),\end{aligned} \tag{1.85}$$

and the perturbed oscillator

$$\mathrm{V}_{\Psi}(\mathrm{x}) \;=\; \frac{1}{2}\sum_{\mathrm{j}}\left(\frac{\mathrm{p}^2}{\mathrm{m}_{\mathrm{j}}} + \mathrm{m}_{\mathrm{j}}\omega_{\mathrm{j}}^2\mathrm{x}_{\mathrm{j}}^2\right) + \sum_{\mathrm{j}}\mathrm{m}_{\mathrm{j}}\omega_{\mathrm{j}}^2\cos\left(\mathrm{x}_{\mathrm{j}} - \mathrm{vt}\right). \tag{1.86}$$

The separate insertion of each of this two potentials into the Schrödinger equation delivers us with a standard solution. But we are more interested in the interaction representation rather than in the Schrödinger representation. A typical part of our information based interaction Hamiltonian H_{I} for a water molecule is:

$$\begin{aligned}\mathbf{H}_{\mathrm{I}} &= \mathbf{N}^{ms}_{H_1O}\boldsymbol{\alpha}\left(m_{H_1},0\right)\boldsymbol{\alpha}^{\dagger}_{O}\left(m_{O_1},0\right)\\ &= i\hbar\int \boldsymbol{\Phi}^{\dagger}_{H_1O}\left(x\right)\boldsymbol{\Phi}_{H_1O}\left(x\right)\Psi^{\dagger}_{H_1}\left(x',0\right)\Psi_{H_1}\left(x',m_{H_1}\right)\\ &\qquad \Psi^{\dagger}_{O}\left(x'',m_O\right)\Psi_{O}\left(x'',0\right)d^3x\,d^3x'\,d^3x''\\ &= i\hbar\int \mathscr{H}_I(x,x',x'')\,d^3x\,d^3x'\,d^3x''.\end{aligned} \tag{1.87}$$

To get the equation of motion e.g. for $\boldsymbol{\Phi}^{\dagger}_{H_1O}$ in the interaction representation the calculation follows the same pattern as before

$$\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{\Phi}^{\dagger}_{H_1O} = \frac{\mathrm{i}}{\hbar}\left[\mathbf{H}_{\mathrm{I}},\boldsymbol{\Phi}^{\dagger}_{H_1O}\right]. \tag{1.88}$$

Such equations have to be solved for all field operators that are part of the interaction Hamiltonian and afterwards inserted in the action integral

$$\mathrm{W}_{12} \;=\; \int_{\mathrm{t_1}}^{\mathrm{t_2}}\mathscr{L}_{\mathrm{I}}\left(\boldsymbol{\Phi}\left(\mathrm{x}\right),\boldsymbol{\Psi}\left(\mathrm{x}\right)\right)\mathrm{dt} = -\int_{\mathrm{t_1}}^{\mathrm{t_2}}\mathscr{H}_{\mathrm{I}}\left(\boldsymbol{\Phi}\left(\mathrm{x}\right),\boldsymbol{\Psi}\left(\mathrm{x}\right)\right)\mathrm{dt}. \tag{1.89}$$

Hereby we pay attention to the fact that the interaction Hamiltonian includes no derivatives therefore we can set $\mathscr{L}_{\mathrm{I}} = -\mathscr{H}_{\mathrm{I}}$. By the calculation of this integral (with even more elaborated formulas for $\mathbf{H}_{\mathrm{I}}$ like (1.77) we can achieve the result that a H_2O molecule is stable or not. In addition we can go on this way and calculate whether the combination of two H_2O molecules is stable and so on. Finally we can calculate whether water can originate or not. The maximal fitness is achieved if not only some stable liquid evolves but this liquid also holds all additional, positive features of water. This maximal fitness can not only be accomplished by the calculations of the action integral but also by explicit calculation of the relevant information as defined in 1.2.5.2.

The symmetry group of the Schrödinger equation and therefore for the field operators $\boldsymbol{\Phi}\left(x\right)$ and $\boldsymbol{\Psi}\left(x\right)$ is the $\mathrm{U}(1)$. This implies that the local gauge transformation for these operators is defined by the phase transformation $e^{\varphi(x)}$ (entirely analogous to classical electromagnetic fields) and $\mathscr{L}_{\mathrm{I}}$ must also be invariant under such arbitrary phase changes (analogous to classical electromagnetic fields). The Lagrangian density mentioned above fulfills this invariance requirement.[2]

1.2.5.5 Appendix

Fermionic anticommutators:

$$\left\{\mathbf{a}_{\mathrm{i}}\left(l\right),\mathbf{a}^{\dagger}_{\mathrm{j}}\left(m\right)\right\} = \delta_{\mathrm{ij}}\delta_{lm},\left\{\mathbf{a}_{\mathrm{i}}\left(l\right),\mathbf{a}_{\mathrm{j}}\left(m\right)\right\} = 0,\left\{\mathbf{a}^{\dagger}_{\mathrm{i}}\left(l\right),\mathbf{a}^{\dagger}_{\mathrm{j}}\left(m\right)\right\} = 0. \tag{1.90}$$

[2] Please remind that we mentioned in this contribution three different symmetries: symmetry of the interaction Hamiltionian, symmetry of the information flow, and symmetry of an organism (body compartments).

Bosonic commutators:

$$\left[\mathbf{b}_{\mathrm{i}}(l), \mathbf{b}_{\mathrm{j}}^{\dagger}(m)\right] = \delta_{\mathrm{ij}}\delta_{lm}, \left[\mathbf{b}_{\mathrm{i}}(l), \mathbf{b}_{\mathrm{j}}(m)\right] = 0, \left[\mathbf{b}_{\mathrm{i}}^{\dagger}(l), \mathbf{b}_{\mathrm{j}}^{\dagger}(m)\right] = 0. \tag{1.91}$$

Here l and m stands for the set of all internal states ("quantum") numbers including the time (commutators respectively anti-commutators are calculated at equal time), the indices i and j are labels of fermionic units (e.g. robot cells); in the case of bosonic operators they characterise the message type (bosonic field). If the commutator of two operators is zero then both equivalent fields (physical terms) can be measured simultaneously.

1.3 Functional and Reliability Modelling of Swarm Robotic Systems

Alan Winfield, Wenguo Liu, Jan Dyre Bjerknes

A robotic swarm is an example of a stochastic, dynamical and often non-linear system. Developing models that allow overall swarm properties to be predicted from the low-level parameters of the individual robots that comprise the swarm is challenging. For this reason many swarm robotics algorithms are validated with reference to simulation studies or limited real-robot experiments only, with no underpinning mathematical model or proof. This approach is inherently limited since simulation or real-robot experiments can only explore small parts of a system's parameter space, and hence provide only weak "inductive" proof of an system's correctness, or reliability. Yet if swarm robotic systems are to find real-world application, especially in safety- or mission-critical applications (Rouff *et al.*, 2003; Truszkowski *et al.*, 2004; Winfield *et al.*, 2006b), we need the strong validation provided by mathematical models of both swarm function and swarm reliability.

This chapter is presented in two parts. In the first section we review approaches for mathematical modelling of collective robotic systems, and outline a macroscopic modelling approach based upon developing a probabilistic finite state machine (PFSM) description of the overall swarm, then expressing the PFSM as a system of differential equations that model the change in the average number of robots in each state, with time. We then illustrate this approach with a case study example of a mathematical model of a wireless connected swarm of mobile robots operated in unbounded space. In the second section we use a modified version of the same case study swarm system to develop failure modes and effects analysis (FMEA), and a reliability model for the swarm. In particular we address the common assumption that swarm systems are automatically scalable and robust and show, for our case study swarm system, that this assumption is incorrect.

1.3.1 Macroscopic Probabilistic Modelling in Swarm Robotics

In recent years probabilistic approaches to modelling swarm robotic systems have been developed and successfully applied. One way to classify these is based on their representation of the swarm and its units.

Microscopic models reproduce each real robot in the targeted system separately, with dedicated — more or less detailed — representations. The *Macroscopic* approach instead models the target swarm robotic system with a single representation, for instance summarising fractions or total numbers of robots in the swarm engaged in specific tasks.

One of the first examples of probabilistic modelling of a swarm of robots at the microscopic level is that proposed by (Martinoli *et al.*, 1999) to study object aggregation; robot's interactions with other robots and the environment are modelled as a series of stochastic events, with probabilities determined by simple geometric considerations and systematic experiments with one or two real robots. The very same microscopic method was applied to the analysis of collaborative stick pulling (Ijspeert *et al.*, 2001).

In general, macroscopic models are more computationally efficient than their microscopic counterparts. One of the fundamental elements of the macroscopic probabilistic model are the Rate Equations, which have been successfully applied to a wide variety of problems in physics, chemistry, biology and the social sciences. For instance, Sumpter and Pratt (Sumpter & Pratt, 2003) developed a general framework for modelling social insect foraging systems with generalised rate functions (differential equations). Sugawara and coworkers (Sugawara & Sano, 1997; Sugawara *et al.*, 1999) first presented a simple macroscopic model for foraging in a group of communicating and non-communicating robots, with analysis under different conditions; for further work see (Sugawara & Watanabe, 2002). (Lerman & Galstyan, 2001; Lerman & Galstyan, 2002b) proposed a more generalised and fundamental contribution to macroscopic modelling in multi-agent systems. In (Lerman & Galstyan, 2002a), they presented a mathematical model of foraging in a homogeneous multi-robot system to understand quantitatively the effects of interference on the performance of the group. In (Lerman *et al.*, 2004), they developed a macroscopic model of collaborative stick-pulling, and the results of the macroscopic model quantitatively agree with both embodied and microscopic simulations. Agassounon and Martinoli used the same approach to capture the dynamics of a robot swarm engaged in collective clustering experiments (Agassounon & Martinoli, 2002).

Rather than using a time-continuous model, Martinoli and coworkers (Martinoli, 2003; Martinoli & Easton, 2003; Martinoli *et al.*, 2004) considered a more fine-grained macroscopic model of collaborative stick-pulling which takes into account more of the individual robot behaviours in the discrete time domain using difference equations. They suggested that time-discrete models are the most appropriate solution for the level of description characterised by logical operators and behavioural states. Similarly, Correll et al. (Correll & Martinoli, 2005) used a macroscopic probabilistic model for analysis of beaconless and beacon-based strategies for a swarm turbine inspection system, and furthermore to find an optimal

collaboration policy minimising the time to completion and the overall energy consumption of the swarm in (Correll & Martinoli, 2006b; Correll & Martinoli, 2006a). In (Correll & Martinoli, 2007), a macroscopic probabilistic model is proposed to analyse self-organised robot aggregation inspired by a study on aggregation in gregarious arthropods.

Both microscopic and macroscopic probabilistic modelling approaches rely on two main common assumptions (Martinoli & Easton, 2003; Martinoli *et al.*, 2004): Firstly, the fulfilment of Markov properties (or semi Markov properties), i.e. the robot's future state depends only on its present state and on how much time it has spent in that state. This assumption is true when robots use reactive control: robots decide on future actions based solely on input from sensors and the action they are currently executing. Therefore the robots can be represented as stochastic Markov processes and the system can be modelled as a probabilistic finite state machine. Secondly, the assumption that the coverage of the arena by the groups of robots is spatially uniform, and the low-level strategies of the robot do not play a critical role on the metric of the system of interest. Indeed, finding an appropriate mathematical description for the transition probabilities is the main challenge in applying both microscopic and macroscopic probabilistic modelling approaches. The second assumption becomes particularly useful for computing the transition probabilities for the robots. In this case, the probabilities of basic events, detecting an object for instance, only depend on geometrical considerations and are given by the ratio of the total extended area of the object related to the total area of the arena where the robots could appear. However, uniform coverage might not always be the case and depends on the environment and the robots' controllers. For example, (Hayes *et al.*, 2000) considered a more complex situation where the distribution of the robot cannot be assumed to be uniform in the arena for an odour plume localisation task. The configuration of the environment and the robots' controllers must be taken into account for probabilistic models.

Despite the success of the above examples, there is little existing work on mathematical analysis of adaptive multi-robot systems in dynamic environments. Lerman and Galstyan (Lerman & Galstyan, 2003; Galstyan & Lerman, 2005; Lerman *et al.*, 2006) extended the macroscopic probabilistic model to study distributed systems composed of adaptive robots that can change their behaviour based on their estimates of the global state of the system. In their study, a group of robots engaged in a puck collecting task need to decide whether to pick up red or green pucks based on observed local information. They claim that the model can be easily extended to other systems in which robots use a history of local observations of the environment as a basis for making decisions about future actions. Liu and Winfield (Liu *et al.*, 2007; Liu, 2008; Liu *et al.*, 2009) developed a macroscopic model for a swarm of foraging robots with adaptation, where the different priorities in behaviour selection and the heterogeneities in individual parameters pose great challenges to modelling. Their model has been successfully used to analyse the performance of the adaptive foraging swarm and further to design optimal parameters for individual controllers.

1.3.1.1 Methodology

For most behaviour-based robotic systems, although the behaviour of a particular robot at a given time is fully determined, the transitions from one state (behaviour) to another could exhibit some probabilistic properties over time within the population of the swarm. The central idea of probabilistic modelling, either micro- or macroscopically, is to describe the experiment as a series of stochastic events and use rate equations to capture the dynamics of these events. A general approach to developing a macroscopic probabilistic model for swarm robotic tasks can be summarised as follows:

step 1. describe the behaviour of the individual robots of the swarm as a finite state machine (FSM);
step 2. transform the FSM into a probabilistic finite state machine (PFSM), describing the swarm at a macroscopic level;
step 3. develop a system of rate equations for each state in the PFSM, to describe the changing average number of robots among states at a macroscopic level;
step 4. measure the state transition probabilities using experiments with one or two real robots, or estimate them using analytical approaches, and then
step 5. solve the system of rate equations.

The controller design for individual robots may take the behaviour based approach introduced by (Brooks, 1986), in which a robot's behaviour is normally determined by its current sensor inputs. A state in the FSM may include one or more of these low level behaviours, and the transition from one state to another will only depend on its current state, rather than historical states; or sometimes on how long the robot has spent in that state. In step 2, the FSM can be transformed into a PFSM by replacing the conditions associated with the transition edges in the FSM with the probabilities that the corresponding conditions are true. In the PFSM, each state represents the average number of robots in that state. The changes of average number of robots in each state of the PFSM over time can then be described using a set of rate equations, either in continuous time or discrete time. Since in a robotics system both analog and digital sensors or actuators are used, we use difference equations (DE) in discrete time to capture the dynamics of the system rather than ordinary differential equations (ODE) in continuous time.

Estimating the transition probabilities for the PFSM (and the DEs) can be a significant challenge, although in some cases these probabilities can be measured by running experiments with one or two real robots; however such an approach is not ideal since the ultimate goal of a mathematical model is analysis and prediction of the effect of individual robot parameters on collective behaviour, rather than trying to match the model to experiments. The transition probabilities are properties of the swarm — they are not design parameters but byproducts of interactions among individuals or between the individual robots and their environment. Using measured values as part of a mathematical model renders the model somewhat less convincing. Moreover, changing environmental conditions or individual parameters

might well then invalidate the measured transition probabilities. In some cases it is not practical to measure all the required transition probabilities because they vary dynamically (time dependent). We advocate geometrical approaches to estimating these probabilities, based on reasonable assumptions. One such assumption is that the distribution of robots within their operational area is, over time, uniform, so the probability that one event is triggered, for example collision with other robots or a bounding wall, could be calculated as the ratio of two areas that the robot covers, on average.

The DEs can be solved analytically or numerically, depending on the complexity of the model. For some tasks, obtaining a direct solution would be possible, for example certain difference equations, in particular linear constant coefficient difference equations, can be solved using z-transforms, see (Martinoli *et al.*, 2004; Agassounon, 2003) for details. For some DEs it is impossible to find a solution using a direct approach such as the z-transform. A numerical approach is then used. In step 5, to obtain numerical solutions for the DEs, certain initial conditions are normally applied, for example the initial number of robots in each state. Another widely used condition is that the total number of robots in each iteration must remain constant.

We now take a simple task as an example to clarify the above approach. Fig. 1.18(a) shows a finite state machine, with only two states, for the robots engaged in a task. Clearly, at each time step, a robot could be in either state A or state B. The robot will transfer from state A to state B whenever the condition a is true. However, it will stay in state B for τ_b seconds and then will move to state A after this time is up. Correspondingly, a probabilistic finite state machine is presented in Fig. 1.18(b). Now the transition between two states in the PFSM is described as a probabilistic event rather than a deterministic event triggered by certain conditions. For example, p in this PFSM represents the probability that condition a is true, which means that a robot will transfer from state A to B with probability p each time step. The probability of transferring from B to state A is 1 but the transfer will be delayed by τ_b seconds, which is the same as in the counterpart FSM. Although the PFSM is derived from the FSM at the individual robot level, it will be used to investigate the changes of number of robots in each state in the swarm by introducing another variable into each state. Here $N_A(k)$ and $N_B(k)$ are used to denote the average number of robots in states A and B respectively. A set of difference equations are then developed to capture the changes of this specific system. Let us consider state A first: the number of robots in state A decreases because of some robots moving to

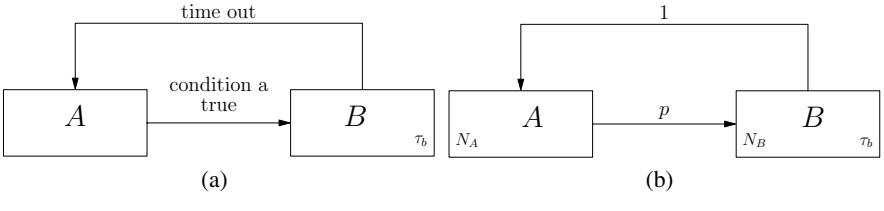


Fig. 1.18 A finite state machine (left) with two states, and its corresponding probabilistic finite state machine (right).

state B, but it increases because of other robots transferring back from state B, thus we have:

$$N_A(k+1) = N_A(k) - pN_A(k) + \Delta_B(k - T_b) \tag{1.92}$$

where $pN_A(k)$ denotes the number of robots moving to state B at time step k, and $\Delta_B(k)$ represents the number of robot transferring to state B at time step k, which is

$$\Delta_B(k+1) = pN_A(k) \tag{1.93}$$

since there is a delay in the transition from state B to A, the number of robots moving to state A from B, at time step k, is equivalent to $\Delta_B(k - T_b)$, where T_b is discretised from τ_b.

Similarly, for state B, we have

$$N_B(k+1) = N_B(k) + pN_A(k) - \Delta_B(k - T_b) \tag{1.94}$$

In fact, since the population of the swarm is constant, say N_0, Eq. (1.94) can be simplified as

$$N_B(k+1) = N_0 - N_A(k+1) \tag{1.95}$$

Assume that the transition probability p has been obtained as outlined above, then given the initial conditions, for example $N_A(0) = N_0$ and $N_B(0) = 0$, the number of robots in each state at any time step, i.e. $N_A(k)$ and $N_B(k)$, can be obtained using a numerical approach. We can therefore use these results to quantitatively analyse the steady-state or dynamic group performance of the system.

1.3.1.2 Case Study: Functionally Modelling a Wireless Connected Swarm of Mobile Robots

Although the simple example above covers the most common type of states and transitions in the macroscopic modelling approach, modelling real systems differs from case to case. The complexity of the model depends very much on the specific task and the metrics of greatest interest. Generally, a number of simplifying assumptions and techniques must be used to model a real system. To illustrate these techniques, a case study is presented in this section. The case study focusses on how to construct a PFSM based on the metrics of interest and how to estimate state transition probabilities in an unbounded environment.

A class of algorithms which make use of local wireless connectivity information alone to achieve swarm aggregation have been developed in (Nembrini *et al.*, 2002; Nembrini, 2005). The basic algorithm, which we refer to as the α-*algorithm*, is very simple. The default behaviour of a robot is forward motion. While moving each robot periodically broadcasts an "I am here" message. The message will of course be received only by those robots that are within wireless range: its neighbours. If the number of a robot's neighbours should fall below the threshold α then it assumes it is moving out of the swarm and will execute a 180° turn. When the number of neighbours rises above α (i.e. when the swarm is regained) the robot then executes

a random turn. This is to avoid the swarm simply collapsing on itself. We say that the swarm is *coherent* if any break in its overall connectivity lasts less than a given time constant. Coherence gives rise to both swarm aggregation and a (coherent) connected *ad hoc* wireless network. In the interests of simplicity we can consider each robot as having three basic behaviours, or states: move *forward* (default); *coherence*, triggered by the number of neighbours falling below α, and *avoidance*, triggered by the robot's collision (proximity) sensor.

The robot updates its connectivity information less frequently than its proximity sensor data. The ratio of connectivity sampling rate to the sampling rate of proximity sensors, which we refer to as *cadence*, is introduced into the basic α-algorithm to prevent the robot from updating its connectivity state too frequently (we need to give the robot time to complete its turn in response to a connection loss, for example, before re-checking its connectivity). By default, the robot will move forward at a fixed velocity. It will update its connectivity state after a certain duration, say T_C (steps), and if it finds the number of connected neighbours has dropped below the threshold α, then it will move into the *coherence* state and execute the U-turn behaviour to try to recover the lost connections; if and when the number of connected neighbours then increases, the robot will execute a random turn. Providing the number of connected neighbours remains at or above α, the robot can lose or gain connections but remain in the *forward* state. Thus, depending upon its connectivity, a robot will either remain in the *forward* state or switch between *forward* and *coherence* states unless it collides with other robots (triggered by the proximity sensor). Such an event will cause the robot to move into the *avoidance* state and execute a collision avoiding turn for time T_A (steps), after which the robot will return to its previous *forward* or *coherence* state. Note that changes in connectivity take precedence over collision avoidance, thus if a change of connectivity is detected while the robot is in the *avoidance* state (i.e. taking avoiding action), the robot will - if required - immediately transition into the appropriate coherence or forward behaviours.

Fig. 1.19 shows the basic robot Finite State Machine (FSM). We reflect the fact that the avoidance behaviours are subsumed within the two top-level states *coherence* and *forward* by showing sub-states $avoidance^C$ and $avoidance^F$. Note that although changes in connectivity take precedence, because the proximity sensor is sampled more frequently than the connectivity (defined above as cadence) collision avoidance is still assured.

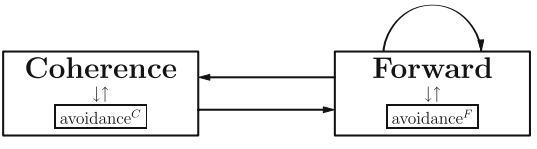


Fig. 1.19 Robot Finite State Machine.

1.3.1.3 A Probabilistic Model of Connectivity

In the α-algorithm $robot_i$ has a number of connected neighbours d_i. Clearly, the range of values for d_i is bounded. The maximum value d_{max} is determined geometrically by the ratio of the areas covered by the wireless sensor range and the avoidance

sensor range; for the robot to remain in the default *forward* state, the lower bound on d_i is α. Now in the α-algorithm when $d_i < \alpha$ the robot moves into the *coherence* state in which it turns back to try and recover the swarm and hence bring d_i back to a value greater than or equal to α. However, the coherence behaviour is not always successful and it is possible for a robot to have fewer than $(\alpha - 1)$ connections. In fact, the robot will continue to try and recover the swarm for values of $0 < d_i < \alpha$. Based on these observations we can now propose a PFSM which completely models the swarm connectivity, as shown in Figs. 1.20 and 1.21.

Fig. 1.20 is, in effect, the simple FSM of Fig. 1.19 expanded to show every possible number of network connections in each of the two states *coherence* and *forward*, together with every possible transition between the states and their probabilities.

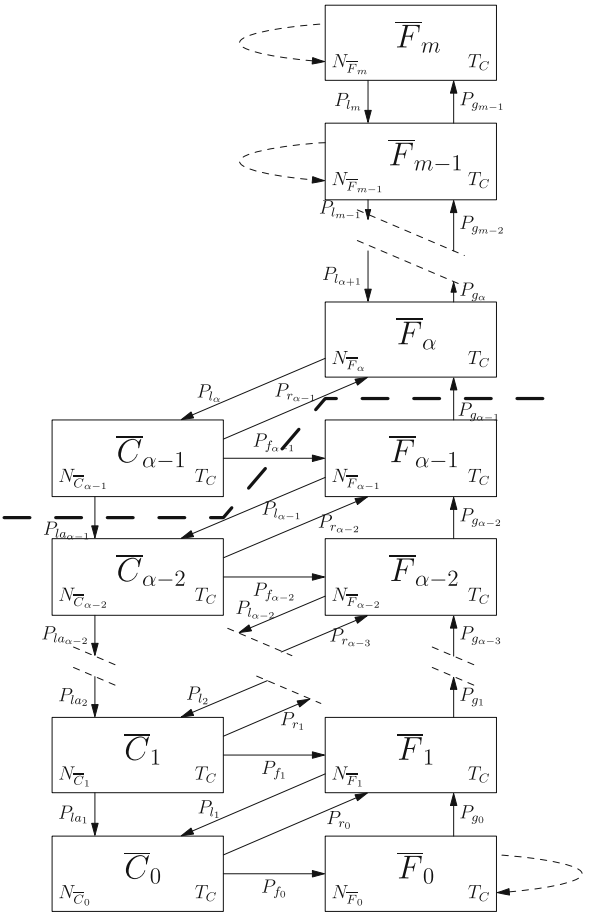


Fig. 1.20 PFSM of the robot controller. $\overline{F}_i$ represents the *forward* state with i connections; $\overline{C}_i$ represents the *coherence* state with i connections. $N_{\overline{C}_i}$ and $N_{\overline{F}_i}$ indicate the average number of robots in corresponding states and T_C indicates the number of time steps spent in each state.

Fig. 1.21 Left: *coherence* state $\overline{C}_i$ expanded to show sub-states A_i^C and C_i. **Right:** *forward* state $\overline{F}_i$ expanded to show sub-states A_i^F and F_i. The average number of robots in each state is shown as $N_\cdot$; T_A is the number of time steps spent in the *avoidance* states A_i^C and A_i^F.

Each of the discrete *forward* states represents a different value of d_i; the $\overline{F}_m$ state is the state with the maximum number of connections d_{max}, the $\overline{F}_{m-1}$ state is the state with $d_{max}-1$ connections, counting down until we reach the $\overline{F}_0$ state with 0 connections; there are a total of $d_{max}+1$ *forward* states, including $\overline{F}_0$. Note that $\overline{F}_0$ is the "lost robot" state representing the failure of the algorithm to maintain the coherence of the swarm. Consider the *forward* state $\overline{F}_\alpha$. The loss of a connection with probability P_{l_α} will cause a transition into the *coherence* state $\overline{C}_{\alpha-1}$. If the action of that state is successful then the robot will transition, after T_C steps and with recovery probability $P_{r_{\alpha-1}}$ back into the $\overline{F}_\alpha$ state. If, on the other hand, the coherence behaviour fails, the robot will move into the $\overline{F}_{\alpha-1}$ state. The likelihood of this is the coherence failure probability $P_{f_{\alpha-1}}$. A loss of connection in each of the forward states $\overline{F}_1 \ldots \overline{F}_\alpha$ will trigger a transition into *coherence* states $\overline{C}_0 \ldots \overline{C}_{\alpha-1}$ respectively.

Fig. 1.21 completes the PFSM by expanding the two states $\overline{C}_i$ and $\overline{F}_i$ into their respective sub-states, again reflecting the structure of the FSM of Fig. 1.19. Fig. 1.21(right) shows that a robot in each of the *forward* states F_i might collide with another robot, with probability P_{a_i}, triggering a transition into its corresponding *avoidance* state A_i^F, returning to the initial *forward* state after T_A steps. Similarly, Fig. 1.21(left) shows that a robot in each of the *coherence* states might also collide with another robot triggering its transition into corresponding *avoidance* states A_i^C, also returning after T_A steps.

1.3.1.4 Geometrical Estimation of Transition Probabilities

Given the PFSM for the $\alpha-$algorithm, a set of difference equations can be derived for the state transitions following the approach outlined in Sect. 1.3.1.1. The full model has a number of transition probabilities, summarised in Table 1.5. Each of these probabilities is conditional on the connectivity status for those robots and therefore impossible to measure in experiments with one or a few real robots.

Here a novel geometry-based approach is developed to estimate these probabilities in our wirelessly connected swarm. Let V denote the normal forward speed of each robot. It follows that the relative speed between two robots varies from 0 to $2V$ and the relative heading varies from 0° to 360°. Consider one of the robots in the swarm, say $robot_i$, with d_i neighbours at time step k. Fig. 1.22 illustrates some of its neighbours, shown as $robot_A$, $robot_B$, $robot_C$ and $robot_D$. Let us assume that $robot_i$ is in either *forward* or *coherence* states, then after one time step (of duration

Table 1.5 State transition probabilities, d_i represents the number of connections for $robot_i$.

probabilities	comments
$P_{a_{d_i}}$	collision with another robot
$P_{l_{d_i}}$	loss of a connection in *forward* state
$P_{g_{d_i}}$	gain of a connection
$P_{r_{d_i}}$	recovery of a connection
$P_{f_{d_i}}$	failure to recover a connection
$P_{la_{d_i}}$	loss of a connection in *coherence* state

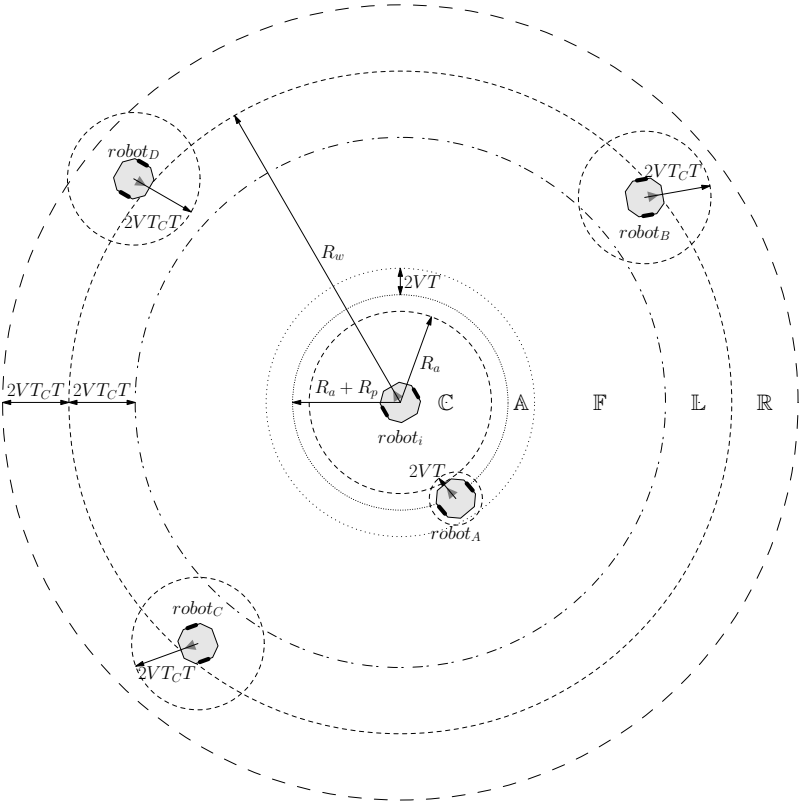


Fig. 1.22 *Robot*$_i$ and its neighbours. Robots are marked with filled circles. Each robot has a communication range R_w and avoidance radius R_a, R_p denotes the physical size (radius) of the robot. $\mathbb{C}$, $\mathbb{A}$, $\mathbb{F}$, $\mathbb{L}$ and $\mathbb{R}$ in the figure represent the collision, avoidance, forward, connection loss and connection recovery areas respectively; each is an annular region bounded within two circles with the same origin.

T), each of its neighbours will move a distance from 0 to $2VT$. It is clear that only the robots close enough will have a chance of moving into $robot_i$'s collision area (within radius R_a, marked $\mathbb{C}$ in Fig. 1.22), and thus drive $robot_i$ to change to state *avoidance*. For instance, as shown in Fig. 1.22, $robot_A$ may possibly trigger $robot_i$'s avoidance sensor next time step while $robot_B$, $robot_C$ and $robot_D$ cannot. Similarly, after T_C steps, $robot_C$ located in area $\mathbb{L}$ will possibly move out of $robot_i$'s communication range resulting in $robot_i$ losing one connection, and $robot_D$ located in area $\mathbb{R}$ might move into $robot_i$'s communication range, with some probability, in which case $robot_i$ will gain one connection at step $k+T_C$. However, $robot_B$ located in area $\mathbb{F}$ can neither trigger $robot_i$'s avoidance sensor nor cause a change in the number of its connected neighbours. Thus, in order to estimate state transition probabilities, we need only consider situations where neighbouring robots fall within the annular regions in Fig. 1.22: $\mathbb{A}$, in which a collision might occur; $\mathbb{L}$, in which a connection loss might occur; or $\mathbb{R}$, in which a connection recovery might occur. A detailed derivation of each of these probabilities is given in (Winfield *et al.*, 2008).

1.3.1.5 Running the Macroscopic Model

We now run the macroscopic model with state transition probabilities estimated using the geometrical approach. Fig. 1.23 shows the average number of robots in states *forward*, *coherence* and *avoidance*, in which we merge states A^C and A^F from the sub-PFSMs in Fig. 1.21, plotted against connectivity. The left-hand plots show the results collected from a sensor-based simulation using Player/Stage, while the right-hand side plots show the results from the PFSM model run with the estimated state transition probabilities. The total average number of robots, summing all states, is also plotted as the topmost curve in each graph.

First we note that the PFSM model generates the same "bell" shaped curves as the simulation, and for all three values of α the peak occurs at or very close to the same connectivity value. The PFSM model for $\alpha = 5$ somewhat underestimates the number of robots in all three states and also shows a longer "tail" of robots with high connectivity values than is measured from simulation; however, the model shows reasonable agreement at very low connectivity values, especially in predicting 'lost' robots (with connectivity of zero). At $\alpha = 10$ the macroscopic model again shows a longer tail of high connectivity robots than the simulation; also evident is the same overestimate in the number of robots in the *forward* state at connectivity values below α. The overestimate in *forward* robots is even more pronounced at $\alpha = 15$. We also see that the 'lost' and very low connectivity robots are not seen in the model for $\alpha = 10$ and $\alpha = 15$. In all three pairs of results the greatest discrepancy between the macroscopic model and the simulation is in robots in the *forward* state. In contrast, the macroscopic model shows much stronger agreement with simulation for the number of robots in *coherence* and *avoidance* states. Given the simplifying assumptions for constructing of the PFSM and estimating of the transition probabilities, it is perhaps surprising that the macroscopic model does generate such plausible results for the swarm connectivity structure. For details of low-level robot parameters and full discussion of the modelling assumptions refer to (Winfield *et al.*, 2008).

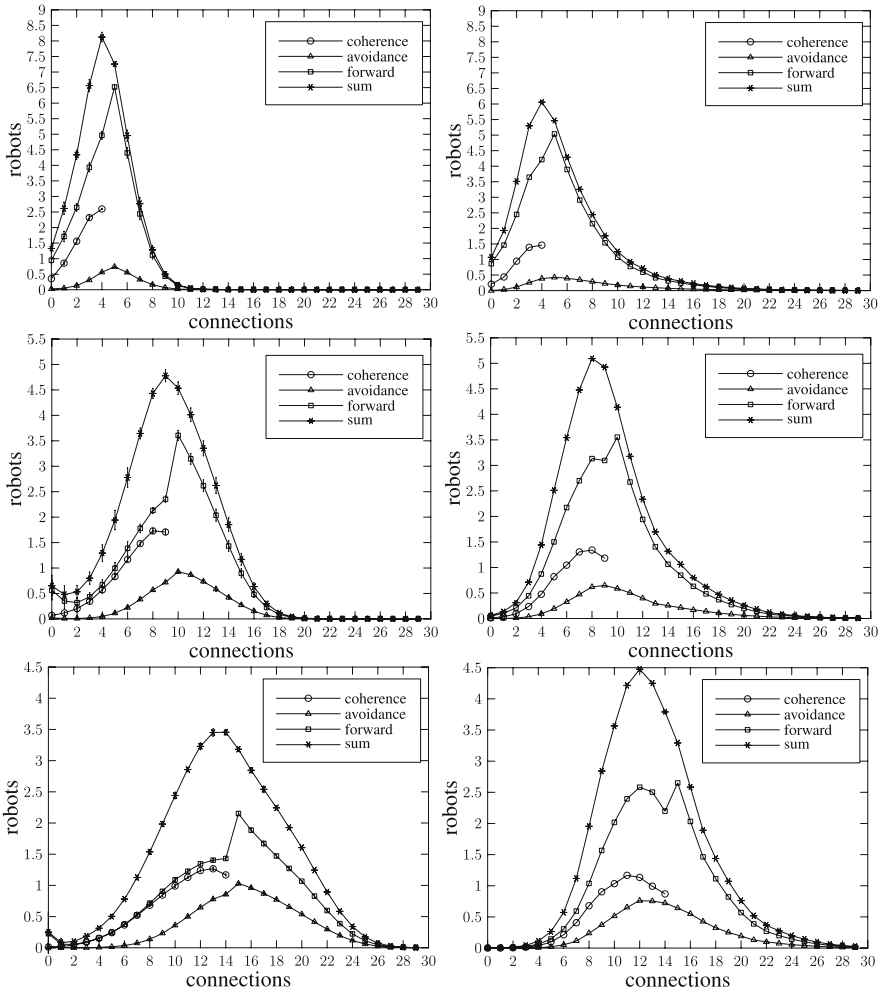


Fig. 1.23 Number of robots in state *coherence*, *forward* and *avoidance* plotted against number of neighbours (connectivity). From top to bottom, $\alpha = 5,\ 10$ and 15. **Left:** Player/Stage simulation, average of 10 runs, each simulation lasts for 10000 seconds. **Right:** Macroscopic model using geometrically estimated probabilities.

1.3.2 Reliability Modelling of Swarm Robotic Systems

Research papers in Swarm Robotics frequently assert that swarm robotic systems are both scalable and robust. We accept the defining criteria for swarm robotics set out in (Dorigo & Şahin, 2004), which include importantly "local and limited sensing and communication", and we note that these criteria also state that swarm robotics studies should "aim for scalability". The fact that individual robots in the swarm make decisions based only on local sensing and communication is assumed to lead

naturally to swarms that will scale to very large numbers of robots. The high degree of parallelism in robot swarms, which typically consist of homogeneous robots, is assumed to lead to a high level of robustness and dependability. It is indeed true that robot swarms can exhibit an unusual level of tolerance to failure of individual robots, or external threats, when compared with conventionally engineered distributed systems, but it is not safe to assume that scalability and robustness are automatically properties of all (or any) swarm systems. It is surprising therefore that, in the field of swarm robotics, there has been very little systematic study of dependability and fault tolerance; but see (Winfield *et al.*, 2005; Winfield & Nembrini, 2006; Christensen *et al.*, 2009).

In this section we develop a reliability model for a case study swarm of robots that exhibit emergent, or self-organised, swarm taxis. We first undertake a failure modes and effects analysis (FMEA) for the case study swarm, then show that we need to model this swarm — from a reliability perspective — as a k-out-of-N system. We then extend the k-out-of-N reliability model to take account of worst-case partial robot failures and swarm scaling properties, introducing the new concept of swarm self-repair. We conclude with a model of reliability as a function of swarm size.

1.3.2.1 Case Study: Reliability Modelling Emergent Swarm Taxis

We modify the α-algorithm described above in Sect. 1.3.1.2 in two ways. Firstly, the *coherence* behaviour is achieved not by making a 180° turn when the number of a robot's connected neighbours falls below α, but instead by timing the duration since the robot last made an avoidance manoeuvre and if that value exceeds a given threshold ω, the robot turns towards its estimate of the centre of the swarm; an estimate based on readings from the ring of infrared proximity sensors around the robot's body. To increase the distance at which robots can sense each other, and also to enable robots to distinguish between robots and ambient infra-red, each of the robots is equipped with infra-red emitters that flash at 80 Hz. By sampling the sensors at 400 Hz and passing the data through a bandpass filter the 80 Hz flashing is reliably detected. Each robot can then estimate the direction of the local centre of the swarm based on which of its sensors detect a flashing signal from other robots. For the results obtained from hardware trials with real robots reported below we set $\omega = 2.5$ s; ω (like α) controls the overall swarm density.

Secondly, we add an additional "beacon" sensor to each robot. The beacon sensor is a very simple sensor, in that it is unable to detect the range and bearing of the remote beacon and has only a two-state output: on = illuminated or off = not-illuminated. An important feature of the physical placement of the beacon sensor is that it can be occluded by other robots, thus those robots that have a direct line-of-sight to the beacon will have beacon sensors illuminated, and those robots that are in the shadow of other robots will have beacon sensors not-illuminated. This means that for a typical swarm only the robots on or close to the leading edge of the swarm (with respect to the beacon) will have illuminated beacon sensors. In our experimental trials we use an IR beacon and make use of the same IR sensors that

are used for both short-range communication and collision avoidance, for beacon sensing.

We then introduce a simple symmetry breaking mechanism. Each robot has short-range avoidance sensors (for sensing collisions with obstacles or, more typically, other robots). We set the avoid sensor radius for those robots that are illuminated by the beacon to be slightly larger than the avoid sensor radius for those robots in the shadow of other robots. This simple mechanism results in a net swarm movement (taxis) toward the beacon. Note that the swarm taxis is an emergent property of the swarm: with a simple two-state beacon sensor a single robot cannot sense the direction of the beacon, and even with the symmetry breaking mechanism two or three robots are not enough to give rise to emergent swarm taxis; experimentally we find that swarm taxis requires at least five robots. This is important to our case study as we are interested in determining the reliability of a swarm with emergent swarm behaviours. For a detailed analysis of the swarm taxis behaviour see (Bjerknes *et al.*, 2007).

1.3.2.2 Failure Modes and Effects Analysis

In this section we undertake a Failure Mode and Effect Analysis (FMEA) for our case study robot swarm. The methodology is straightforward, see (Dailey, 2004). We attempt to identify all of the possible hazards, which could be faults in robots or robot sub-systems (internal hazards), or environmental disturbances (external hazards). Then, in each case, we analyse the effect of the hazard on each of the overall swarm behaviours. Thus, we build up a picture of the tolerance of the swarm to both types of hazard and begin to understand which hazards are the most serious in terms of compromising the overall desired swarm behaviours. FMEA is, at this stage, essentially qualitative. Here we consider only internal hazards; external hazards (i.e. communications noise) were investigated in (Nembrini, 2005).

First we identify the internal hazards. In keeping with the swarm intelligence paradigm our robot swarm contains no system-wide components or structures, thus the only internal hazards that can occur are faults in individual robots. Since, in our case, the robots of the swarm are all identical, then (internal) hazards analysis requires us to consider only the faults that could occur in one or more individual robots, and then consider their effect on the overall swarm behaviours. Table 1.6 identifies the fault conditions for an individual robot.

Table 1.6 Internal Hazards for a single robot

Hazard	Description
H_1	All systems failure
H_2	Coherence sensor(s) failure
H_3	Avoidance sensor(s) failure
H_4	Beacon sensor failure
H_5	Motor sub-system failure

Table 1.6 makes the assumption that failures of robot sub-systems can occur independently. This is a reasonable assumption, given that our mobile robots are in reality an assembly of complex but relatively self-contained sub-systems. Hazard H_1 represents a total failure of the robot; failure of the robot's power supply would, for instance, bring about this terminal condition. Hazards H_2, H_3 and H_4 represent a failure of the robot's communication, avoidance and beacon sensing functions respectively. Finally hazard H_5 motor failure, covers the possibility of mechanical or motion-controller failure in one or both of the motors in our differential drive mobile robot, such that the robot is either unable to move at all or can only turn on the spot (which from an overall swarm point of view amounts to the same thing).

Let us now consider the effects of each the hazards enumerated above on the overall swarm behaviours. We will consider here the effect on the overall swarm of the hazard occuring in one or *a small number* of the individuals in the swarm. Of course the question of how many is a small number in this context is important, and will return to the question of what proportion of robots need to fail in order to *seriously* compromise the desired overall swarm behaviours.

Hazard H_1: total systems failure. Complete failure of one or a small number of robots caused, for instance, by power failure will clearly render the robot(s) stationary and inactive. They will be wirelessly disconnected from the swarm and will be treated, by the swarm, as static obstacles to be avoided. Ironically, given that this is the most serious failure at the level of an individual robot, it is relatively harmless as far as the overall swarm is concerned. Apart from the loss of the failed robots from the swarm, none of the overall swarm behaviours are compromised by this hazard - however, we can expect the swarm to be temporarily slowed by the obstacle represented by the failed robot. We therefore label this effect E_1, with an upper-case E to denote that it is a potentially serious fault.

Hazard H_2: coherence sensor failure. Failure of the coherence sensors sub-system in one or a small number of mobile robots means that those robots cannot sense the centre of the swarm. Given that basic swarm aggregation depends upon the coherence mechanism, then robots with fault H_2 might become physically lost to the swarm and wander off at random. As far as the swarm is concerned these robots simply become (moving) obstacles to be avoided. The overall swarm behaviours are, however, essentially unaffected. This hazard has, therefore, a relatively benign effect, except of course that the failed robots remain mobile within the environment and - in some circumstances - this may be undesirable. We label this effect e_2, with a lower-case e to denote that it is a non-serious fault.

Hazard H_3: avoidance sensor failure. Failure of the avoidance sensor(s) in one or a small number of robots has little effect on overall swarm behaviour. A single robot with failed avoidance sensors will be avoided by the other robots in the swarm and hence have no overall effect. In the unlikely event that two or more robots with

Table 1.7 Summary of Failure Modes and Effects

Swarm Behaviour	H_1	H_2	H_3	H_4	H_5
Aggregation	-	e_2	-	-	-
Ad hoc network	-	e_2	-	-	-
Beacon taxis	E_1	e_2	-	-	E_5

failed avoidance sensors collide with each other then physical damage might result from such collisions, but the overall swarm behaviours remain unaffected.

Hazard H_4: beacon sensor failure. Failure of the beacon sensor in one or a small number of robots has a practically undetectable effect on the overall swarm behaviour. This is because the emergent beacon taxis behaviour results from the symmetry breaking mechanism outlined above. Since a differential is created between two substantive parts of the swarm, the effect of one or a small number of robots with failed beacon sensors is negligible.

Hazard H_5: motor failure. The effect of motor failure in a single robot, or small number of robots, is interesting. Robot(s) with fault H_5 become – in effect – stationary but, given that their signalling and other sensing systems continue to function, they remain within the swarm. These robots continue to fully contribute to the swarm aggregation and *ad hoc* network emergent behaviours. It is only when the swarm needs to physically translate its position. i.e. for the beacon taxis behaviour, that hazard H_5 becomes a serious problem. In this case robots with motor failure will have the effect of physically anchoring the swarm, either impeding or, at worst, actually preventing the swarm from moving toward its target. This is a potentially serious hazard since one or a small number of robots with motor failure could seriously compromise the desired swarm-taxis behaviour. We shall label this fault effect as E_5, with an upper-case E to denote that it is potentially serious.

To summarise, Table 1.7 shows the swarm fault effects, as defined above, generated by one or a small number of robots with hazards $H_1 ... H_5$, for each emergent swarm behaviour. Table 1.7 clearly shows that the serious swarm failure effects E_1 and E_5 occur in only 2 out of 15 possible combinations of robot hazard and swarm behaviour; 10 out of the 15 hazard scenarios have no effect at all on swarm behaviour, and the remaining 3 have only minor, non-serious, effects.

1.3.2.3 The k-out-of-N Reliability Model

The purpose of a reliability model is to enable the estimation of overall system reliability, given the (known) reliability of individual components of the system, see (Elsayed, 1996). Reliability R is defined as the probability that the system will operate without failure, thus the unreliability (probability of failure) of the system, $P_f = 1 - R$. In our case the overall system is the robot swarm and its components are the individual robots of the swarm.

From a reliability modelling perspective a swarm of robots is clearly a parallel system of N components (robots). If the robots are independent, with equal probability of failure p, then the system probability of failure is clearly the product of robot probabilities of failure. Thus, for identical robots, $R = 1 - p^N$. p can be estimated using a classical reliability block diagram approach on the individual sub-systems of the robot; since the individual robot does not internally employ parallelism or redundancy then its reliability will be modelled as a series system, giving p less than the worst sub-system in the robot, which is most likely to be its motor drive system.

However, this simplistic modelling approach makes a serious and incorrect assumption, which is that the overall system remains fully operational if as few as one of its components remains operational. This is certainly not true of our case study swarm. The desired emergent swarm behaviours require the interaction of multiple robots; our swarm beacon taxis behaviour is a dramatic example: with one robot only the behaviour simply cannot emerge. It is a general characteristic of swarm robotic systems that the desired overall swarm behaviours are not manifest with just one or a very small number of robots. However, the question of how many (or few) robots are needed in order to guarantee a required emergent behaviour in a particular swarm and for a particular behaviour is often not straightforward.

Thus, from a reliability perspective, we need to consider the swarm as a k-out-of-N:G system. That is, a system of N parallel elements which requires that at least k of these elements are operational ("good") for the overall system to function correctly. In a swarm of N robots, if more than $N - k$ fail, the self-organised functionality of the overall swarm will be compromised.

In a k-out-of-N:G system, the probability that *at least* k out of N robots are working at a given time t is given by:

$$P(k,N,t) = \sum_{i=k}^{N} \binom{N}{i} (e^{-t\lambda})^i (1 - e^{-t\lambda})^{N-i} \tag{1.96}$$

where $\lambda = \frac{1}{MTBF}$, (Kuo & Zuo, 2002). MTBF is the mean time before failure of an individual robot.

Based on Eq. (1.96) we can now plot swarm reliability against time for our case study swarm. Experimental trials indicate that at least five robots have to be working in order for the emergent swarm taxis behaviour to work properly. Thus, we can model our swarm as a 5-out-of-N system. Consider now the individual robots' MTBF. Carlson et al. tracked failure data for 13 robots by three different manufacturers over a period of two years. They found the MTBF to be eight hours (Carlson & Murphy, 2003). Experiments with the e-pucks used in our experimental trials might suggest that their failure rate might be higher (because of poor design of the e-puck battery connector). However, as no systematic data is available, the value reported by Carlson et al. will be used here. Fig. 1.24 (top) plots Eq. (1.96) for a swarm of ten robots, and shows that the swarm reliability starts to decline rapidly after 100 minutes of operation.

Fig. 1.24 (bottom) plots the reliability of the same swarm of ten robots, with the same values for k and MTBF, against the distance the swarm will travel

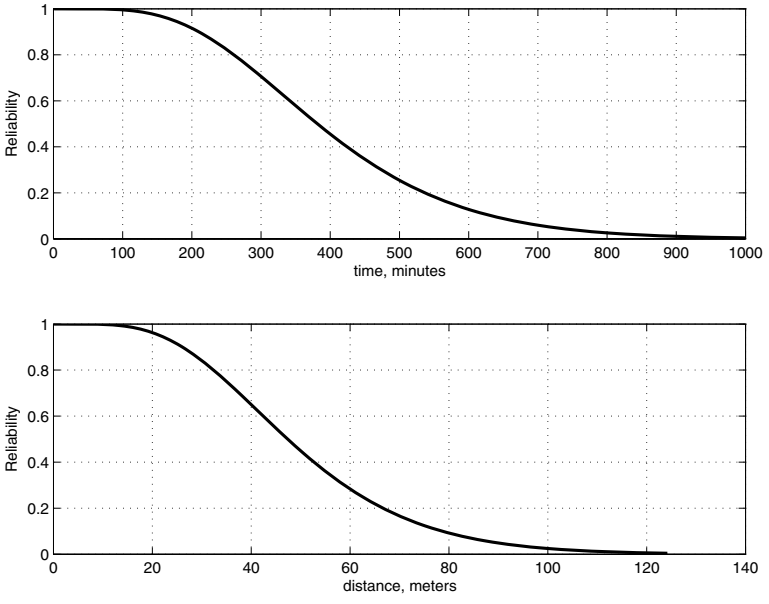


Fig. 1.24 Top: The reliability of a robot swarm modelled as a k-out-of-N system, with $k = 5$, swarm size $N = 10$ robots and MTBF = 480 m. Bottom: Reliability of the same swarm as a function of distance travelled, based on a measured mean swarm velocity of 12.4 cm. per min. for a swarm of 10 robots.

(the emergent swarm taxis behaviour) based on a measure mean swarm velocity of 12.4 cm per minute for a swarm of 10 robots. Although providing some insight, the reliability assessments based on the k-out-of-N model here fail to take into account two important factors. Firstly, each robot that fails is likely - depending on the exact nature of that failure - to slow down the swarm; if the failed robot(s) are immobile then the swarm will slow down until it "escapes" from the failed robots, leaving them behind. Secondly, the swarm velocity might then change after then failed robot(s) have been left behind, typically a smaller swarm (of at least 5 robots) will have a higher swarm taxis velocity. We now analyse these factors in more detail in order to improve the swarm reliability model.

1.3.2.4 Swarm Self-repair

We now introduce the concept of *swarm self-repair*. Consider the case-study swarm and its failure modes and effects analysis outlined above in Sect. 1.3.2.2. We have conducted a series of trials of the emergent beacon swarm-taxis algorithm, using 10 e-puck robots (Mondada *et al.*, 2009), in which we artificially introduce different failure modes into one or more robots of the swarm. Our trials broadly confirm the FMEA and demonstrate that, while all failure modes have the effect of slowing down swarm progress toward the beacon, the swarm is tolerant to the simultaneous

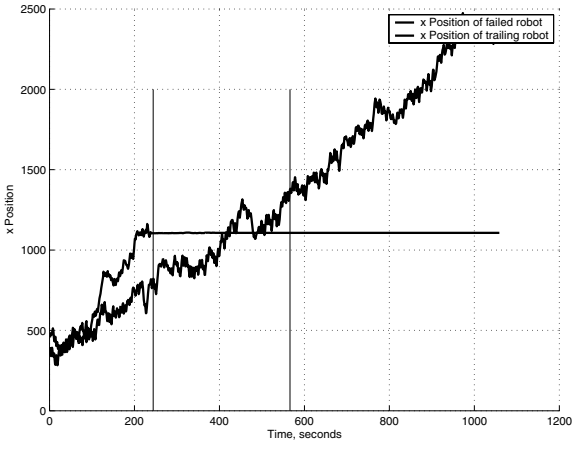


Fig. 1.25 Hardware trials using 10 e-puck robots: single robot complete failure H_1, swarm self-repair time. Two robots are tracked: the failed robot and the trailing robot from the rest of the swarm. At about 250 s. a single robot on the leading edge of the swarm experiences failure H_1; at about 580 s. the trailing robot escapes the failed robot.

(i.e. worst case) failure of more than one robot. Furthermore, we notice two different categories of effect on the overall swarm: (1) sensor failures $H_2...H_4$ which slow down progress of the swarm, but the whole swarm reaches the beacon and (2) motor failures H_1 and H_5 which hold back progress of the swarm until the swarm breaks free of the failed robots; for a detailed analysis of these results see (Bjerknes, in press). Consider the second, and more serious category, which gives rise to the notion of swarm self-repair.

Refer to Figs. 1.25 and 1.26. We define swarm self-repair time as the time between (simultaneous) motor failure of one (or more) robots and the point at which the trailing robot in the rest of the swarm escapes the influence of the failed robot(s). This is a useful metric because it varies with both the type of robot motor failure (H_1 or H_5) and the number of robots. Table 1.8 lists the measured swarm self-repair times for one and two simultaneous failures for failure modes H_1 (robot completely failed) and worst case H_5 (robot partially failed - motors failed but electronics still operational). For comparison the table also shows a baseline notional self-repair time: the time the swarm would take to leave behind a failed robot if that robot failure did not slow down the swarm.

1.3.2.5 Swarm Scaling and Reliability

We have argued that in the k-out-of-N reliability model above, the minimum value of $k = 5$ because the swarm taxis property is present even with as few as 5 robots. For $N = 10$ robots and an MTBF of 8 hours, this reliability model suggests that the swarm will become unreliable after approximately 100 minutes. While it is clear that we can increase the swarm reliability by increasing the individual robots' MTBF,

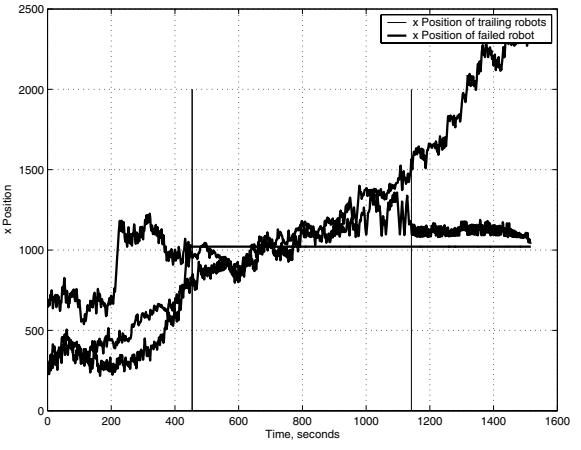


Fig. 1.26 Hardware trials using 10 e-puck robots: single robot partial failure H_5, swarm self-repair time. Three robots are tracked: the failed robot, the trailing robot from the rest of the swarm and a third healthy robot left behind with the failed robot. At about 450 s. a single robot on the leading edge experiences failure H_5; at about 1150 s. the trailing robot of escapes the failed robot. Observe that one healthy robot is also left behind by the swarm and only 8 robots proceed to the beacon.

Table 1.8 Mean swarm self-repair times for the case study swarm of $N = 10$ e-puck robots. Ten runs for each case. *In the final case of two partially failed robots, in only six runs did the swarm reach the beacon.

Case	Mean (s)	Std. Dev. (s)
Baseline (no penalty)	328	174
One failed robot H_1	387	132
Two failed robots H_1	453	172
One failed robot H_5	879	417
Two failed robots H_5	1279	see note*

can we also make the swarm more reliable by increasing swarm size? At first it might seem plausible to suggest that the increased redundancy in a larger swarm would maintain reliability for a longer period. One may even be led to believe that the swarm could be made reliable for an arbitrarily long time, given a sufficiently large number of robots. This is not correct, and we now combine a model of swarm self-repair with the k-out-of-N model to determine the maximum upper size for our case-study swarm.

Consider the argument informally. When a swarm is larger it will take longer to self-repair than a smaller swarm. There are two reasons for this. Firstly, it is a property of our case study swarm that the swarm taxis velocity reduces with increasing swarm size. Secondly, the swarm is physically larger and must move a longer distance before it is fully self-repaired. Thus the self-repair rate will *remain constant*

with increased swarm size. However, for a given robot MTBF, the swarm failure-rate will *increase* for larger swarms. It is unavoidable that at some point the failure rate will overtake the self-repair rate of the swarm, and the swarm will come to a complete halt - the desired emergent swarm-taxis property will fail. In fact a swarm of sufficient size would die under its own weight, so to speak, before it has even started to move.

We now estimate the values of k and self-repair time t_s as a function of N. Thus the k-out-of-N model written as Eq. (1.96) will be modified to take the time to self-repair into account.

The value of k

In experimental tests it is clear that, for complete failures H_1, two out of ten robots could fail without permanently damaging the swarm. The swarm would always self-repair. The cases with partial failure H_5 fared less well. When one out of ten robots failed, the swarm did always self repair, even though a functioning robot might occasionally become stuck with the failed robot. But when two out of ten robots failed, the swarm would suffer a complete breakdown in four out of ten cases, and in the remaining six cases, as many as three healthy robots stayed behind with the failed robots.

Based on this the value of k will be conservatively estimated as 90% of N for a k-out-of-N:G system. In other words, when the swarm has ten percent failed robots or less it will be assumed that it can self repair. Arguably, this may not hold true for larger swarms - the empirical evidence is limited to swarms with ten robots. But this is our best estimate from the evidence available.

The value of t_s

We know from an analysis of the scaling properties of our case study swarm (Bjerknes, in press), that swarm-taxis velocity v as a function of N follows this relationship:

$$v(N) = CN^{-\frac{1}{2}} \tag{1.97}$$

Where C is a scaling constant. Thus larger swarms move more slowly. Note, as stated already, that the minimum value of swarm size N for the swarm to exhibit swarm taxis is 5, thus Eqn. 1.97 is not valid for $N < 5$.

Clearly, the diameter d, of the swarm will increase with swarm size.

$$d(N) = D\sqrt{N} \tag{1.98}$$

Where D is the density constant for the swarm.

Since a robot can fail anywhere within the swarm: on the leading edge, in the middle of the swarm or at the trailing edge, the average distance that the swarm needs to move before it has moved away from the failed robot will be half the diameter, $\frac{d}{2}$. Thus the self-repair time becomes $t_s = \frac{d}{2v}$.

Thus,

$$t_s(N) = \frac{D\sqrt{N}}{2C\frac{1}{\sqrt{N}}} \tag{1.99}$$

Which simplifies to

$$t_s(N) = \frac{D}{2C}N \tag{1.100}$$

Eq. (1.100) is important as it demonstrates that the self-repair time increases linearly with N. Based on this equation it is now possible to introduce a new constant for a given swarm, namely the self-repair-time-constant. Let this constant have the symbol S for Self-repair, where $S = \frac{D}{2C}$. Now we have established that S is linear with N, we can determine its value experimentally. For a swarm with ten robots with one partially failed robot the mean self-repair time was found to be 879 s (see Table 1.8). This was for a case with ten robots, so the self-repair constant for our case study swarm, for the worst case partial failures H_5, then becomes $S = \frac{879}{10} = 87.9$.

Using the estimated values for k and t_s and the k-out-of-N reliability model we can now plot swarm reliability against swarm size N.

Fig. 1.27 shows that with an MTBF of 8 hours, a swarm with as few as 40 robots will have a reliability of only 0.5. This reliability model is based on a number of assumptions (including, for instance, a circular swarm morphology that remains constant with increasing swarm size), together with experimentally estimated constants. Notwithstanding these assumptions and estimates, the main idea that the self-repair-time increases with larger swarms is well argued based on the experiments presented here. Even though the actual reliability for a given swarm size may be a somewhat

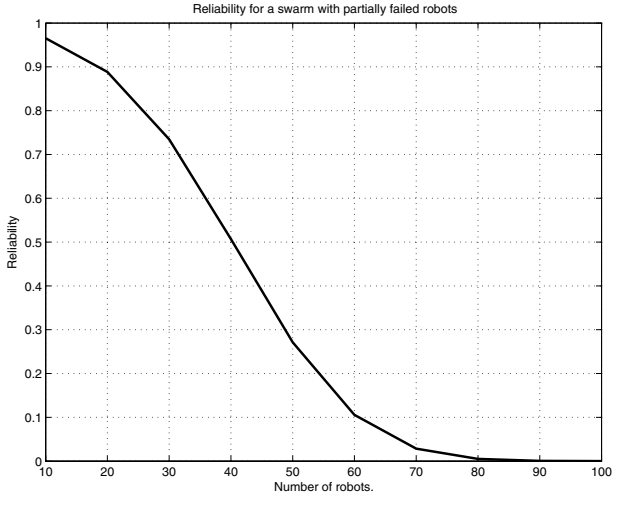


Fig. 1.27 Reliability of the case study swarm as a function of swarm size, based on a k-out-of-N reliability model and assuming worst case partially failed robots H_5; $k = 0.9N$, self-repair-time-constant $S = 87.9$ and robot MTBF 8 h.

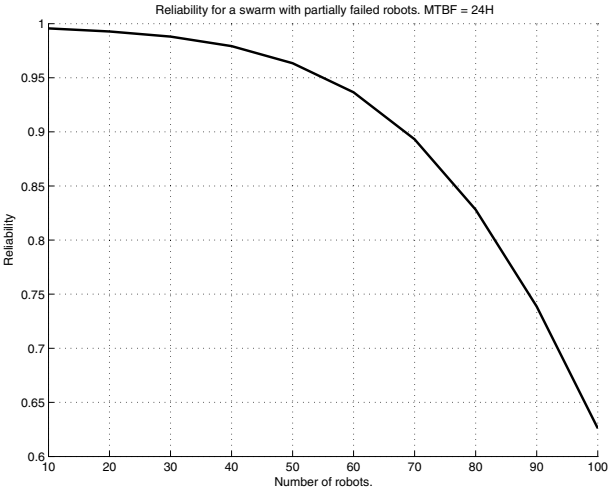


Fig. 1.28 Reliability of the case study swarm as a function of swarm size, based on a k-out-of-N reliability model and assuming worst case partially failed robots H_5; $k = 0.9N$, self-repair-time-constant $S = 87.9$ and robot MTBF 24 h.

higher or lower than the k-out-of-N model suggests, it is undoubtedly true that our case study swarm will eventually become non-functioning with increasing size, and that this occurs at a much lower swarm size than one might intuitively expect. Clearly we can significantly improve swarm reliability by increasing robot MTBF, as shown in Fig. 1.28, for a four-fold improvement of individual robot MTBF to 24 h.

1.3.3 Concluding Discussion

In Sect. 1.3.1 we have reviewed approaches for mathematical modelling of collective robotic systems, and outlined a macroscopic modelling approach based upon developing a probabilistic finite state machine (PFSM) description of the overall swarm, then expressing the PFSM as a system of differential equations that model the change in the average number of robots in each state, with time. We then illustrated this approach with a case study example of a mathematical model of a wireless connected swarm of mobile robots operated in unbounded space. The model demonstrates a novel robot-centric approach for estimating state transition probabilities. There is no doubt that proving the correctness of collective systems requires mathematical modelling, and we believe that the macroscopic probabilistic approach outlined here provides us with a powerful modelling technique.

Consider now the applicability of this functional modelling approach to symbiotic multi-robot organisms — the subject of this volume. Sect. 1.1, Table 1.4 identifies swarm-mode, organism-mode and the transitions between these modes.

The PFSM modelling approach can be applied in swarm-mode during searching/resources localisation (i.e. foraging) and during the transition from swarm to organism (morphogenesis). We have already developed a PFSM model for adaptive foraging, see (Liu *et al.*, 2007; Liu, 2008; Liu *et al.*, 2009). For morphogenesis the model will be used to predict the time robots take to self-assemble into a 2D planar structure, i.e. the energy cost of this phase. The aim here is to use the model to improve the behaviour of individual robots and hence optimise energy consumption and achieve faster transition from swarm-mode to organism-mode.

Sect. 1.3.2 has shown that the frequent assumption that collective systems based upon the swarm intelligence paradigm are automatically scalable and robust is unsafe. By undertaking a reliability analysis of a swarm system in which the desired swarm properties are truly emergent (self-organising), we have shown that, for the worst-case partial-failures of individual robots, overall system reliability falls very rapidly with increasing swarm size. When compared with conventionally designed distributed systems our case study swarm does exhibit an unusual level of tolerance to failure of individual robots (and indeed a self-repair mechanism) that — in a sense — comes for free with the swarm intelligence paradigm, but that fault tolerance does not scale well. Of course our case-study swarm has no mechanisms for actively identifying and compensating for partially-failed robots, which leads to the conclusion that scaling collective systems from tens to hundreds or thousands of robots might not be achievable without such mechanisms, i.e. distributed artificial immune systems (see Sect. 4.4).

Consider now the applicability of the reliability modelling approach to symbiotic multi-robot organisms. We would argue that the multi-state k-out-of-N approach can be applied to the searching/resource localisation (foraging) task during swarm-mode. It is likely that foraging will require swarm aggregation/taxis so that a subset of robots moves together to an object of interest thus, depending on the algorithm design, the reliability model given in this chapter may apply in some modified form. The k-out-of-N reliability model might also be applicable to organism-mode since, in principle, we might expect a multi-robot organism comprising N robots to continue to function if at least k of these robots are functioning. Clearly the failure modes and effects analysis (FMEA) will be more complex since the consequences of a failed robot, and the type of failure, will depend on the position and function of that robot within the organism.

Chapter 2
Heterogeneous Multi-Robot Systems

2.1 Reconfigurable Heterogeneous Mechanical Modules

Kanako Harada, Paolo Corradi, Sergej Popesku, Jens Liedke

The mechanical characteristics and functionalities of individual robots in a collective symbiotic system are of the utmost importance in order to confer suitable capabilities to the symbiotic robot organisms. However, this does not necessarily mean that the design of individual robots has to be particularly complex from a mechanical point of view. On the contrary, excessive complexity can lead to several disadvantages in the assembled state of the organism, e.g. higher risk of failures and higher electrical and computational power demand. In addition, considering the manufacturing phase of the individual robots themselves, complexity would lead to high development and assembling costs; this is an issue particularly relevant when a large multi-agent symbiotic system is targeted. Finally, considering miniaturized robots, there are severe volume constraints at the design level that may prevent the possibility to integrate complex mechanisms. Consequently, as a rule of thumb, the individual robots of a large collective symbiotic system can be designed to offer the minimal mechanical functionalities able to allow the symbiotic robotic organism to assemble and develop all those *collective configurations* and *reconfiguration strategies* that let specific *collective functionalities* emerge. That's inevitably a compromise choice in the design.

As introduced in Sect. 1.1, a symbiotic robot organism can be seen as the physical evolution of a swarm system of individual robots into a structural system of connected robots. From this "structural" perspective, the mechanical functionalities of the individual robot could correspond to the behavioral rules of the agents in a swarm system that arises collective emergent behaviors. The mechanical interactions between the robots assembled in the organism expand consequently the collective capabilities of the system to a structural dimension.

On the base of the above considerations, it is clear how the design of suitable mechanical features of the individual robots represents a critical issue. In particular, the robot-to-robot connection mechanisms (docking mechanisms) and the

P. Levi and S. Kernbach (Eds.): Symbiotic Multi-Robot Organisms, COSMOS 7, pp. 79–163.
springerlink.com

mechanical degrees of freedom implemented in the individual robots deserve a deep investigation.

From a design viewpoint, it is difficult to decide for the optimal compromises of mechanical features (namely inter-robot docking systems and degrees of freedom) in order to have specific advantages and capabilities both for the individual robots and the symbiotic organism. A comparable problem occurs when one wants to assign specific behavioral rules to agents in swarm systems in order to "program" desired collective behaviors: the path(s) to the task solution is, indeed, not predefined but emergent (Bonabeau *et al.*, 1999), thus it is hard to infer which basic behavioral rules have to be assigned to the individual robots in order to obtain a specific swarm behavior. Faithful modeling and computer simulations could possibly guide in the investigations of the main organism configurations and corresponding capabilities on the base of a particular mechanical choice done for each individual robot, and consequently give some feedbacks in order to suitably modify the mechanical elements. However, that is a hard and complex way to follow. It will evidently appear to an expert reader that the efforts required by tuning the physical characteristics of the mechanical elements, accordingly to kinematics and dynamics simulated in a virtual environment, are much larger than modifying software-based behavioral rules as it has been done for developing swarm intelligence algorithms. The general approach in the mechanical design of the individual robots ends consequently to be more heuristic. The performances and structure characteristics of the individual robot tend to be optimized for its own individual functionalities while the basic capabilities are guaranteed as a modular part of the assembled organism. This can lead to multiple design solutions as described in the following paragraph.

2.1.1 A Heterogeneous Approach in Modular Robotics

The design of each individual robot as a stand-alone unit inevitably ends to favor specific functional characteristics such as locomotion capability, actuation power and robustness, and this can result in multiple design solutions. This is true especially for miniaturized individual robots because focusing on one feature means finally to degrade or loose other features due to obvious space constraints. As a consequence of the above mentioned issues, the design process can follow different paths:

- To try to merge the best features of all the conceived designs into a unique individual robot design by accepting performance compromises of the collective system while making the control of the organism easier. We refer to such a system as collective *homogeneous* system. This is the path mostly followed by state-of-the art modular and reconfigurable robotics ((Yim *et al.*, 2003), (Castano *et al.*, 2000), (Christensen, 2006), (Zykov *et al.*, 2007a), (Kamimura *et al.*, 2005), (Salemi *et al.*, 2006), etc.).
- To consider having two or more different individual robot types where each robot is optimised for specific functions. Each robot can assemble into a symbiotic organism by means of compatible docking units, thus empowering the global

capabilities of the collective system in detriment of more complex control of the symbiotic organism due to its heterogeneity. We refer to such a system as a collective *heterogeneous* system as introduced in (Lyder *et al.*, 2008).

- To integrate "tool modules" with the above mentioned collective *homogeneous* system. Tool modules can be generally defined as devices whose functions are dedicated to a specific task. The tool modules can simply dock with the assembled organism, receive commands from the organism and possibly send data to the organism. These tool-modules could be, for instance, wheels, sensors, grippers, etc. By following this path, the system has to accept poor integration of the robot in favor of versatility. This approach is considered to be the evolved version of the collective *homogeneous* system as demonstrated in (Zykov *et al.*, 2008).
- To integrate "tool modules" with the above mentioned collective *heterogeneous* system. The main structure of the organism is composed of two or more different individual robots and the organism can be equipped with "tool modules". The heterogeneity of the system becomes high, making the control more complex. The system is the most versatile and robust to the environment and given tasks. This is a rather new approach in the modular robotics as studied in (Bordignon *et al.*, 2008), (Kernbach *et al.*, 2008b), (REPLICATOR, 2008-2012).

Taking inspiration from the biological domain, it could be observed that natural swarms are often heterogeneous not only for the different behavioral specialization of each swarm member but also from a strict physical viewpoint (e.g., in a same colony there are insects with different physical capabilities, e.g. in ant colonies). However, differently from natural insect swarms, the conceived collective system should also be able to reach a collective structural level. This goal can be more complicated with heterogeneous individual robots, regarding the assembly process itself and, even more, for what concerns the onboard software (e.g., the self-learning and behavioral control of the symbiotic organism).

A final decision about the approach to choose can be critical in terms of capability and performance of the multi-robot system and the final symbiotic organism(s). However, it should be mentioned that it depends also on the specific application the multi-robotic system is intended to. Currently, most of the collective robotic systems are mainly thought for exploration or surveillance, nevertheless some more targeted applications can be found in literature, e.g. in biomedical applications (Harada *et al.*, 2009). A heterogeneous robotic system with tool modules is likely to offer more advantages under specific applications requiring complex and diversified tasks and/or a large interaction with the environment. The heterogeneous robotic system with tool modules is possibly beneficial in terms of the collective system efficiency also from a point of view of the operation carried out by the stand-alone robots; in such a case, each robot can contribute with its own optimized functions in achieving the goals.

For example, it is possible to consider the operation of a heterogeneous collective system with tool modules in exploration scenarios. A basic heterogeneous approach could rely on the investigation of two or more different types of individual robots having compatible docking units. As a case study, two individual robots, namely a

Scout robot and Backbone robot, and one tool module, namely Active wheel, will be described hereafter and shown later in the chapter:

- A "scout" robot equipped with far-range sensors and above all specialized in fast and flexible locomotion that can be used for inspection of the environment and for swift gathering of robots for the assembly. For this purpose, wheeled/caterpillar-like locomotion is advantageous, in particular where challenging terrains have to be engaged. Actuators for the 3D actuation within the organism is mandatory but less powerful actuators are sufficient. It is because the scout robots can be useful when they are docked to the end of a leg or arm of the organism to scan the environment.
- A "backbone" robot, strong in main actuation and stiff in design. The main purpose of this robot is to work as a part of the organism, therefore the casing is strong to provide high stability and the main actuator is able to lift several docked robots to perform 3D motion. The space for 2D locomotion is limited due to the large main actuator, but the 2D locomotion drive is capable of necessary movements for assembly and docking. In addition, the design of the robot allows to use the single DOF of the main actuator for either bending or rotation of the docked joint. Therefore, the powerful actuation is available for any joint in the assembled organism.
- An "active wheel" module as a tool module. Tool modules are optimised for specific functions and designed in a way to compensate aforementioned deficits of the individual robots. The Active wheel, for example, provides the ability to move omnidirectional, lifting and carrying heavy loads (i.e. other robots or organisms) and at the same time is able to provide an additional energy source. This tool can act in standalone mode as well as in organism mode.

The prototypes of the Backbone robot, the Active wheels and the Scout robot are shown in Fig. 2.1.

Following the general issues introduced above, several technical key aspects has to be taken in consideration in the mechanical design of the individual robots. These aspects are mainly related to:

- Integration and miniaturization;
- Locomotion mechanisms;
- Docking mechanisms and strategies;
- Mechanical degrees of freedom.

All these aspects will be discussed in the following also by using the examples of the Scout robot and Backbone robot. The design strategy of the tool module and the example of the Active wheels is described in Sect. 2.1.6.

2.1.2 Integration and Miniaturization

A swarm robot is intended to be part of a large group, therefore especially production costs should be as low as possible to build groups of hundreds or thousands of these robots. A small and simple design could significantly reduce costs, on the

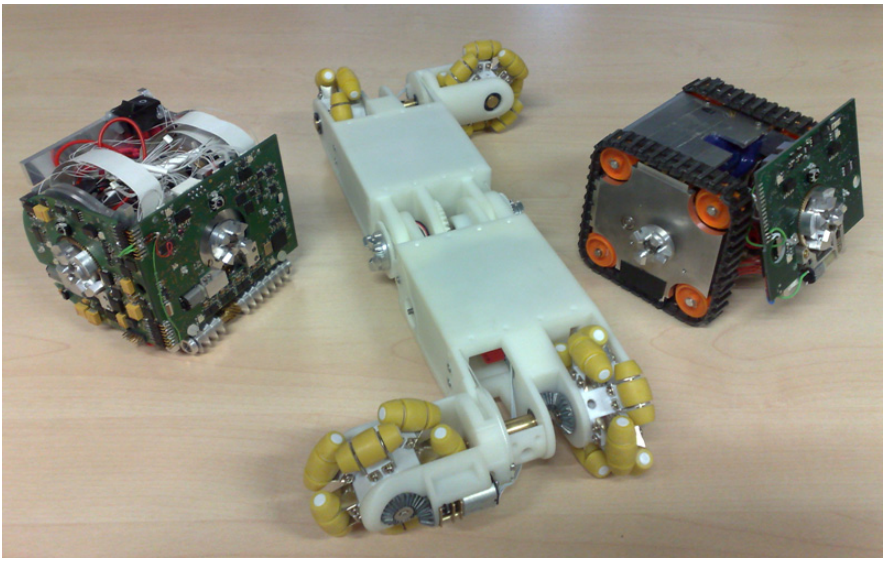


Fig. 2.1 First prototypes of robot designs (from left to right): Backbone robot, Active wheel, and Scout robot.

other hand several key features are required by the individual robot which can be difficult to be integrated into tiny robots. One example for this problem is the energy source required by an autonomous swarm robot. State of the art accumulators can store electrical energy which provide a runtime of several hours, but their dimensions are too big to fit into small robots. This problem get even worse, if the robot shall provide high torque and/or a high velocity.

Since individual robots can be part of an organism and they are used to change the topology of it, some kind of high power actuator is usually used. This kind of motor also requires space together with the battery pack, affecting the size of the individual robot significantly. For a prototype, especially one used in a research project, other features should be considered concerning integration and miniaturization as well. Since only a few dozen robots can be produced in the beginning, standard components like gears and wheels should be used to keep the cost low and accelerate manufacturing of the robots. In addition, it should be noted that the power output of electric motors increase with their size. If the other parts are not too small, standard machinery can be used as well. Miniaturization therefore should not be the key point in designing the robot if one is not aiming especially for small robots.

Another point is the integration of all the components and subsystems into the robot. Each individual robot is equipped with different kind of sensors, actuator(s) for 3D configuration of the organism and 2D locomotion as a stand-alone robot, battery packs and docking units, microprocessors and motor drivers, and all of those need to be assembled in the final state of manufacturing. If the robot is very small, the difficulty level for this task also increases significantly.

The design of the SYMBRION/REPLICATOR robots therefore aim mostly on functionality then on size. The goal is to build a robot capable of moving around, perform actuation inside an organism and provide as many different sensors and high calculation power as possible.

2.1.3 Locomotion Mechanisms

The locomotion capability allows the individual robots to be active in the environment, carrying on tasks of exploration, for instance. The locomotion capability is evidently fundamental when docking with other robots is necessary in order to reach the symbiotic state. Several approaches can be followed for the design of locomotion mechanisms, depending on the requirements that the individual robots and the symbiotic organism have. In classical modular robotics, the individual robot or module has been considered as a part of the modular system, thus it does not have any mechanisms that let it move as a stand-alone system. Instead, locomotion has generally been considered as a capability of the assembled robot and achieved by means of coordinated actuation among the docked modules in order to realize snake-like locomotion, legged-base walking, etc. This can limit the exploration capability of the whole system to the assembled state. In other words, individual robots or modules need to be manually positioned and docked before initiating the operation. When additional modules are requested by an assembled robot at the operation site, the assembled robot needs to go back to a specific zone where individual modules are deployed, or another assembled robot needs to be formed to reach the operation site. Hence, it is a natural consequence to try to devise individual locomotion solutions on each individual robot. This would guarantee the collective system much higher independence, versatility and flexibility. The system can be autonomous and robust especially in an unknown environment where the number of required robots and appropriate topologies of the organism can be determined after the robots reach the operation site.

From the functional viewpoint, locomotion required for the individual robot of a collective symbiotic system can be categorized into three types of locomotion; fast locomotion for the exploration on rough terrains, rather slow locomotion for aggregation, and precise locomotion for the docking alignment. Each individual robot needs to be equipped with these features, but with different priorities. Taking the example of the above mentioned two robotic designs, i.e. the Scout robot and the Backbone robot, the Scout robot has much priority on the exploration task with fast locomotion, while the locomotion for aggregation and docking alignment is much more important for the Backbone robot. For this reason, a tracked locomotion for the Scout robot and an omnidirectional drive for the Backbone robot were chosen for the prototypes, considering the functionalities unique to each robot.

Tracked locomotion is adequate for the quick locomotion on rough terrains. The Scout robot with tracked locomotion is capable of going up a slight slope, climbing over small obstacles, passing over a small hole, and also moving in soft ground. The long-range sensors on board can be used to scan the obstacles around then to navigate the organisms (Fig. 2.2(a)). When the tracked robots are docked together,

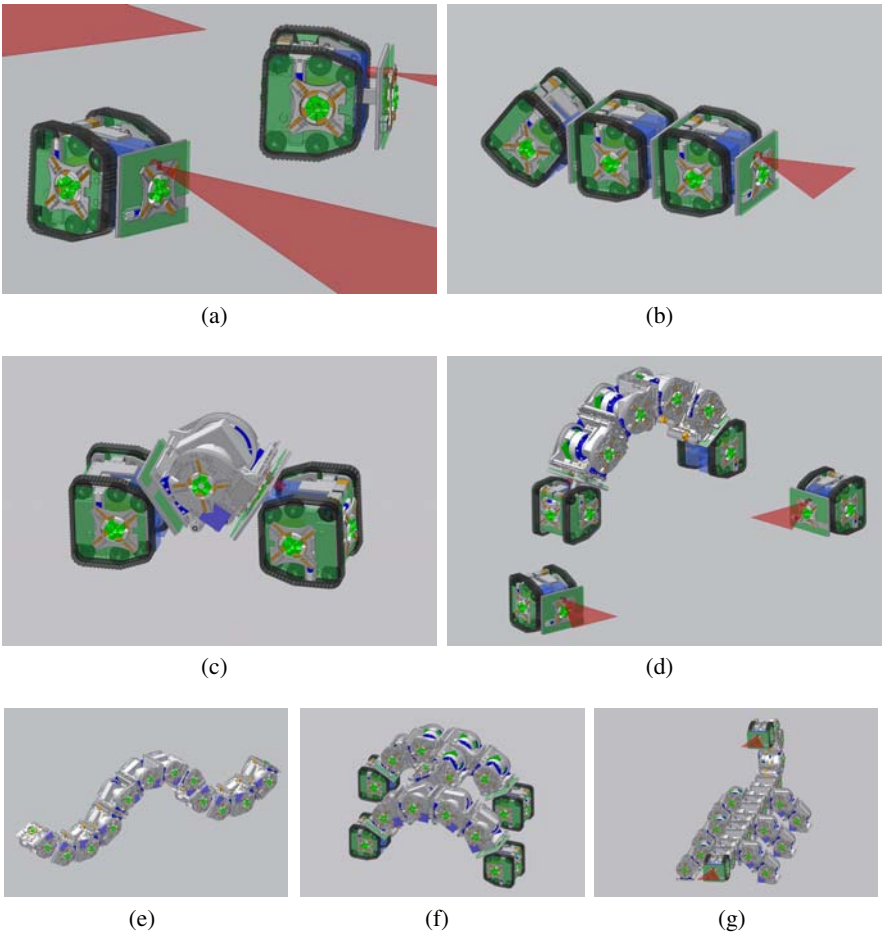


Fig. 2.2 Scout robots: **(a)** Scout robots exploring the surface and guiding the organisms; **(b)** Connected Scout robot; **(c)** Scout robots carrying a Backbone robot; **(d)** Scout robots carrying a chain composed of the Backbone robots. **(e)** Snake shape of an organism; **(f)** 4-legs shape of an organism; **(g)** Scorpion-like organism.

the assembled robot becomes more robust to the roughness of the terrains as shown in Fig. 2.2(b). This high locomotive capability also allows the Scout robots to carry the Backbone robot(s) (see Figs. 2.2(c)(d)). The Backbone robots can form an arm or a leg of an organism in advance, then be carried to the operation site so as to save the energy for 3D actuation in the organism. Thus, the Scout robots are adequate to be "feet" of the organism thanks to its robustness and locomotive capability. The disadvantage of the tracked locomotion is the non-holonomic drive characteristic that hinder efficient docking procedures between the robots.

Regarding the locomotion capability of the Backbone robot, easy assembly of the organism is of utmost importance. Therefore an omnidirectional drive is best since

it offers optimal performance to move to a predefined position under a defined angle. This is important because each individual robot provides at least four different docking units and all of them can be used to form the structure of the organism. Every docking unit need to be reached, regardless of the orientation of the robot which wants to dock. Unfortunately, the integration of an omnidirectional drive requires a lot of space due to the general construction of omnidirectional wheels. Nevertheless, if one look at the details of the docking procedure, complete omnidirectional driving characteristics are not required for the Backbone robot, since the orientation of the robot is predefined by the docking units and therefore only certain directions of movement are necessary. In general, the Backbone robot needs to be able to move forward, backward and to turn since these are the minimum requirements for a swarm robot. Furthermore, under the condition of docking orthogonal to the normal drive direction of the robot, it need to move sideways. A locomotion drive unit which can provide the features of a differential drive plus the possibility to drive to the side is therefore sufficient. Both features are provided by the screw drive, which is used within the Backbone robot. The screw drive locomotion unit itself can be build very small since only two driving motors are required and the driving screws have cylindrical shapes. Beyond the normal use of the nearly omnidirectional drive of the Backbone robot, the screw drive provides the organism with a possibility to move sideway when the screws of all robots within the organism are synchronised. This can be a very helpful feature if a caterpillar like organism need to steer to the side, for example.

An example of a system composed of reconfigurable heterogeneous mechanical modules, i.e. the Scout robots and the Backbone robots, are shown in the Figs. 2.2(e)-(g). All individual robots and organisms work as autonomous standalone systems.

2.1.4 Docking Mechanisms and Strategies

The docking mechanisms are of primary importance in modular robotics as well as in symbiotic multi-robot organisms. They should assure docking and undocking between individual robots, as well as electrical continuity for power sharing and signal transmission. Furthermore, the docking mechanism should tolerate at a certain degree misalignments of individual robots during the docking process (Salemi *et al.*, 2006). Nilsson et al. have investigated design of a docking unit and summarized desirable connector properties (Nilsson, 2002). In this section, the properties required for docking mechanisms are investigated and a guideline for the docking design is proposed. *Docking* is composed of several phases, and each phase has several requirements to be satisfied.

1. **Approach.** The approach of the docking units can be categorized into three modes. The first is the approach of the two locomotive individual robots. Because both robots can move freely, the approach of the docking units is rather easy. The second is the approach of an individual robot to an organism. In this case, the individual robot should be precisely steered. When the individual robot

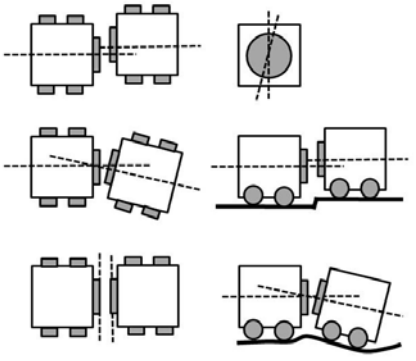


Fig. 2.3 Modes of misalignments.

with non-holonomic locomotion capability needs to be docked to the organism, the docking units on the side walls are not available unless the organism itself can approach the individual robot. Thus, the aggregation of an organism must be carefully planned considering the locomotion capability of the individual robots. The last one is the approach of the two assembled robots or two arms/legs of an organism, and this is especially important for a reconfiguration of the organism.

2. **Alignment.** Misalignment of the docking units fall in one of the six modes of mis-alignments illustrated in Fig. 2.3 or a combination of them. Docking design that allows robust self-alignment is crucial for autonomous assembly of a modular robot. Ground roughness need to be taken into consideration for the docking of locomotive individual robots. In addition, it must be noted that the accuracy of the fabrication and assembly of each robot have strong influence on the alignment accuracy.
3. **Docking and Locking.** A docking unit with hermaphroditic feature is preferable to make the assembly plan easier. The docking must be tight and stable, and the electrical connection between the docked robots must be ensured. In some existing docking designs, the docking is secured by an additional locking mechanism. A simple docking/locking mechanism occupying small space and being actuated with little energy is preferable as well.
4. **Sustainment of the docked status.** The docking status must be sustained without or with minimum power supply. The docking status needs to be independent from the actuation of the assembled robots, otherwise, the additional control is necessary to maintain the docking status.
5. **Unlocking and Undocking.** Another important feature is the capability to allow undocking between two docked robots in case of emergency. If one of the individual robots undergoes failure or malfunction, the robot must be removed from the organism by the other robots. Therefore, it is preferable to undock the robot by activating only one of the docked units.
6. **Separation.** The individual robots need to be separated and move away from the assembled robot after being undocked so as not to hinder following procedures. When an individual robot with non-holonomic locomotion cannot move away

after being undocked, the organism needs to move away from it or another robot needs to come to move it away.

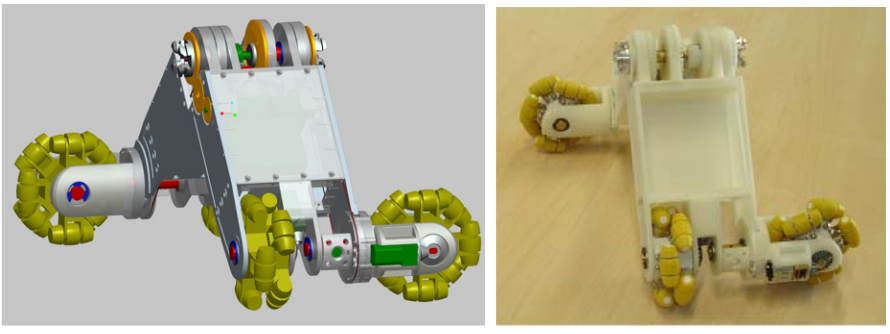
In addition to the above mentioned requirements, easy and low-cost manufacturing for mass production and easy maintenance are important especially when a large multi-agent symbiotic system is targeted. Because multiple docking units are required for an individual robot, the cost of the docking unit is important.

2.1.5 Mechanical Degrees of Freedoms: Actuation for the Individual Robot and for the Organism

Degrees of freedoms in the individual robot set its capability to change the morphology of the symbiotic organism and ultimately strongly influence the capability of the organism to perform complex tasks. In addition, the DOFs can be exploited for tasks to be carried out by the individual robot itself or to achieve a different geometric morphology that is functional to perform specific tasks (e.g., different locomotion strategies). Unfortunately, space constraint and energy issues inevitably limit the implementation of many DOFs on board the individual robot. In the heterogeneous system, each individual robot can have different DOFs based on its role in the system. As for the above mentioned Scout and Backbone robots, the Scout robot can have more than 1 DOF with small actuation force to constitute a small organism for the intensive sensing during exploration or to be docked to a leg of an organism, while the Backbone robot needs strong actuation force to constitute the main backbone of a big symbiotic organism. Hence, in the prototype design, the Scout robot is equipped with 2 DOFs of smaller torque and the Backbone robot has 1 DOF of high torque. Because the docking units cannot be attached to all surfaces, the relation of the actuation axes and the available docking planes must be carefully considered. To the authors' best knowledge, there is no model that can give an optimised combination of the actuating direction(s) and the docking planes, and only a simulation could handle such a large number of DOFs. However, the swam intelligence and self-learning algorithms might not require optimised mechanical solutions as long as all constraints are properly defined.

2.1.6 Tool Module: Active Wheel

In a heterogeneous system, robots of different design can form an organism together. The two individual robots, namely Scout robot and Backbone robot, have been proposed as basic elements to constitute an organism. The design of this individual robot is a result of compromise to integrate all mechanical and electronic functionaries into one robot. The features of such individual robots have to be redundant to be adaptable in an unknown environment. The idea of implementing tool modules into the heterogeneous system is to provide a few specially designed tools to compensate deficits of the individual robots. The design of tool modules needs to be optimized for specific tasks such as sensing with a special sensor, manipulating



(a) (b)

Fig. 2.4 Autonomous tools; **(a)** CAD image; **(b)** Real prototype.

an object, supplying power to the organism and carrying the individual robots or an organism quickly. The individual robots need to share external dimension to be a part of organism and for easy reconfiguration, and they need to be equipped with common electronics, while a tool module can has any shape as long as it can be docked to other individual robots or an organism. As an example of tool modules, we developed a tool module to carry individual robots, named Active Wheel (see Fig. 2.4). This tool module is intended to carry some individual robots quickly from one place to another without using their energy.

The Active Wheel is an autonomous tool robot that is compatible with other two individual robots platforms (Scout robot and Backbone robot) and used for assistance goals. An Active Wheel consist of two symmetrical arms connected in the middle by a hinge (see Fig. 2.4(a)).

This structure gives the opportunity of banding this tool in both directions up to $\pm 90^\circ$ and hence can drive on top and bottom site. Such a symmetrical design does not require distinguishing between bottom and top or between front and rear side. An additional advantage of such geometry is the uniform weight distribution which is important for stable locomotion. Even if the robot is in a skew position a or b it tilts autonomously back into a stable position a_1 or b_1 (Fig. 2.5). Of one of the major tasks of this tool robot is to carry a certain number of individual robots efficiently from one place to another. This condition can be fulfilled only if the Active Wheel can move omnidirectional. Therefore, two omnidirectional wheels are used on each side on the robot. Such kind of wheels have been already proved in many robotics projects e.g. in the RoboCup (Zweigle *et al.*, 2009). Each wheel consists of many small single rolls which are arranged perpendicularly to the driving axle. This assembly allows an active movement in the driving direction of the wheel and simultaneously allow a passive movement in the normal direction. Each of these wheels is driven by a gear motor. Corresponding sensors which are placed on the driving axle detect the rotation speed of the motor. Those are necessary in order to provide complex manoeuvres such as driving curves or other complex trajectories.

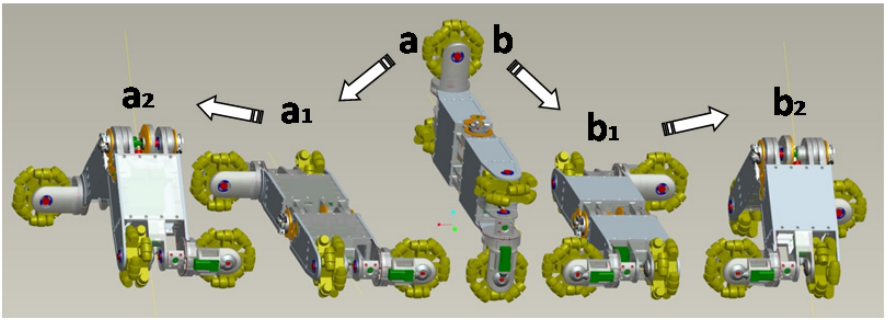
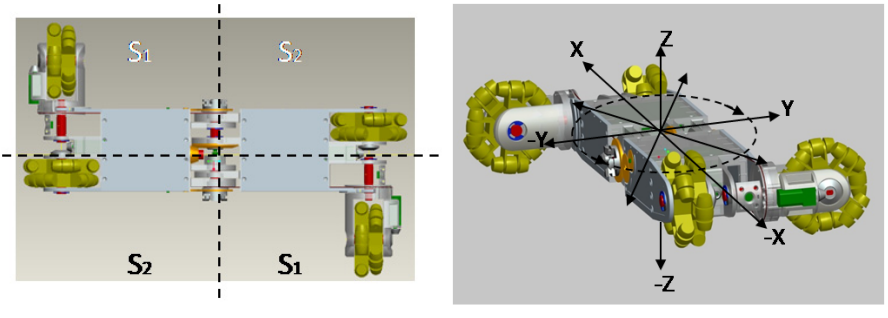


Fig. 2.5 Symmetry and stability of the robot and capability to bend upwards or downwards.



(a) (b)

Fig. 2.6 Autonomous tools; **(a)** Symmetry ; **(b)** Degree of freedom.

The docking between Active Wheel and another robot requires also a very precise control of the wheels.

Additionally to the motor control unit, the Active Wheel is equipped with similar electronic units and components like in the Scout or in the Backbone robot. These comprise for example similar processors, power management, IR sensing unit, ZigBee module, camera etc. All these electronics is mainly required in order to navigate and to launch other robots autonomously and at the same time allow acting as stand-alone robot and fulfill many different tasks in robot swarms. In a stand-alone mode, Active Wheels can be used for getting out of damaged modules or modules that are not be able to move. One possible scenario how Active Wheel can act as a stand-alone robot, is shown in the Fig. 2.7. Two of the Active Wheels are placing a module that was flipped over again in the right position.

As an example of a simple organism, topology of three robots can be considered Fig. 2.8. The idea of this configuration is based on a combination between advanced computational and sensor features, provided by these two individual robots, and a fast movement, provided by Active Wheel. Additionally, the Active Wheel can

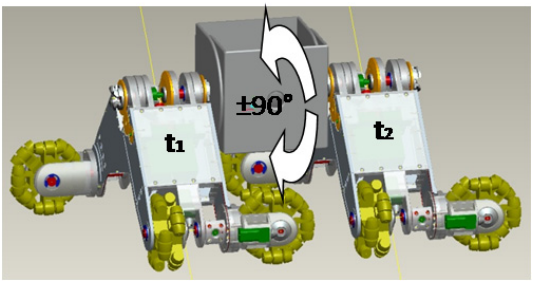


Fig. 2.7 Two Active Wheels carry a defective element.

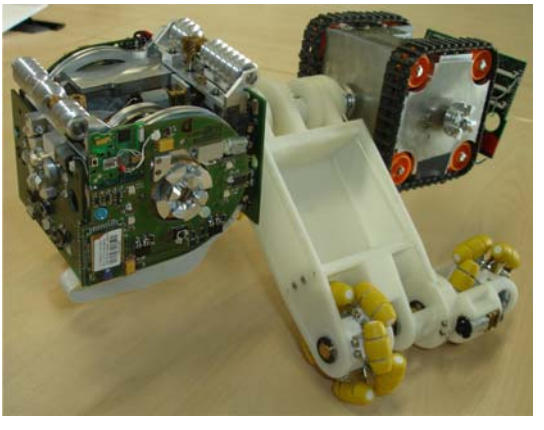


Fig. 2.8 Simple organism - Active Wheels with two different docked modules.

supply both individual robots with extended energy source. As a common system, these three platforms complement each other and demonstrate commonly very outstanding characteristics. Features of a common system essentially overstep capability of each of these individual robots – this is typically collective approach.

2.1.7 Summary of the Three Robotic Platforms

Based on the above discussion, the requirements and solutions of the Scout robot, the Backbone robot and the Active Wheel have been defined as shown in Table 2.1.

To summarize this section, we have to point out two essential issues: integration with electronics, considered in Sect. 2.2 and a need of software protection from mechanical damages, caused during evolving different controllers, see Sect. 4.1. Both issues are essential in a successful design and stepwise improvement of mechatronic platforms.

Table 2.1 Scout robot, Backbone robot and Active Wheel: requirements and solutions

	Scout robot		Backbone robot		Active Wheel	
	Require.	Solutions	Require.	Solutions	Require.	Solutions
Alignment	Rough	Tracked locomotion	Accurate	Omni-directional drive	Accurate	Omni-directional drive
Ground Surface	Rough	Tracked locomotion	Plain	nearly Omni-directional drive	Plain	Omni-directional drive
Locomotion after docking	Required to carry a robot	OK (Locomotion on three surfaces)	Not required	wheels still available for driving	Required to carry	OK (Locomotion on two surfaces)
Speed of locomotion	High	12.5 cm/s	Low	6 cm/s	High	31 cm/s
DOFs of actuation	2 DOF	Bending: $\pm 90^\circ$ Rotation: $\pm 180^\circ$	1 DOF	Bending/ Rotation: $\pm 90^\circ$	2 DOF	Bending/ Rotation: $\pm 180^\circ$
Torque	Low	3Nm	High	up to 7N	High	up to 5Nm
Speed of actuation	Low 30°/s	37.2°/s	High	180°/s	Low	50°/s

2.2 Computation, Distributed Sensing and Communication

Eugen Meister, Oliver Scholz, Jaouhar Jemai, Jiri Havlik, Wenguo Liu, Salah Karout, Guoqiang Fu, Serge Kernbach

Besides the mechanical platform of a robot, that determines its shape and actuation capabilities, a key part of a robot's hardware is its electronic circuitry. It defines resources provided to software, e.g. the size of code and/or data memory, computation speed, communication bandwidth, etc. Processing power and communication pathways dictate how sophisticated the software can be and how much time certain algorithms will require. From a control perspective, too lengthy processing and/or communication time will even make a robot useless, i.e. not capable to operate in real time. For instance, if a robotic organism should start to topple, a controller plus the involved actuators should be fast enough to regain its stability. In addition, sensors are required in order to evolve the robot and to determine its own state. A communication backbone among the different modules of an aggregated organism is mandatory for self-reconfigurable robots, in particular when processing power is to be shared among the organism members.

In conventional, non-reconfigurable robots, the shaping of the electronic's circuitry can already be quite demanding. In reconfigurable robots, however, this task becomes even more difficult since, ideally, the electronics should be designed to be highly modular, i.e. a stand-alone robot that docks to a second should turn into one entity, one organism, together with its fellow robot(s). Thus, a fast and reliable communication channel between these robots is wanted so that both can contribute to processing by task sharing.

Task sharing is very often performed even in stand-alone robot modules by incorporating multiple processors, each serving for a dedicated purpose such as sensor data acquisition, sensor fusion, actuation control, high level behavioural control, etc. This approach has the advantage that it can simplify the design due to a more modular structure. Moreover, computation speed can be dramatically increased since multiple computational tasks are performed in parallel with no need to constantly interrupt an on-board processor. Furthermore, this approach allows parts running independently and increases the robustness of the software. When smartly implemented, one processor may even take over the role of another in the case of a failure. Another benefit is a scalability of processing power, so that individual processors can be set to sleep when not required in order to save energy.

This section first gives some insight into the computational and communication architecture of state-of-the-art reconfigurable robots in Sect. 2.2.1. Then, in Sects. 2.2.2 and 2.2.3 the electronic architecture of SYMBRION/REPLICATOR modules is presented, followed by consideration of general sensor capabilities. After this the distributed on-board vision system, the chosen wireless communication scheme, the employed localisation system and integration issues are described in more details in Sects. 2.2.4-2.2.7.

2.2.1 Electronic Architectures in Related Works

In the following, a few examples of hardware architectures of state-of-the-art reconfigurable robots are given. It is though difficult to directly compare these since every project has its own emphasis, e.g. some are intended for self-reconfiguration, others depend on manual reconfiguration. However, most – if not all – reconfigurable systems employ distributed computation and control. For instance, the ***Molecube*** from Cornell University, New York, is equipped with two ATMEL INC. *ATmega16* microprocessors plus a smaller *ATmega8* embedded in the robot's servo (Zykov *et al.*, 2007b). For more processing power, e.g. required for behavioural control, a dedicated processor module lacking actuation but incorporating a more powerful ARM OLIMEX *LPC-H2148* can be manually attached to the ***Molecubes***. All three on-board controllers are connected to a TTL-level RS232 bus, running at up to 1 Mbits/s. A similar bus is used in addition to provide inter-module communication when multiple modules have been joined.

Superbot of the University of Southern California incorporates two 16 MHz *ATmega128* microcontrollers. Taking account the specific mechanical platform, which consists of 2 half-modules, each of the controllers is responsible for the actuators

and sensors of its half-module in which it is located. However, one of the processors in addition takes care of the behavioural algorithms. Therefore, this microcontroller is referred to as a "master" controller whereas the other as "slave". Both microcontrollers communicate via an I^{2}C-bus[1] at 400 kbits/s. For the inter-modules communication, a 230 kbits/s RS232 bus is being used, with each three of the docking ports of a half-module having its own RS232 interface. In turn, the corresponding microprocessor of a half-module is attached to the group of docking ports via a 1 Mbit/s SPI-bus. Wireless inter-module communication when not docked is performed via an IrDA[2] similar infrared communication scheme with a speed of 230 kbits/s (Salemi *et al.*, 2006).

The ***M-Tran II*** platform from the National Institute of Advanced Industrial Science and Technology (AIST), Japan, employs a *Neuron chip* (ECHELON CORP.) as a "master" CPU plus three microcontrollers (*PIC16F873* and *F877*). The four on-board controllers communicate via a two-wire asynchronous serial bus. Inter-module communications make use of RS-485 and a LON[3]-Protocol (Kurokawa *et al.*, 2003). The successor, ***M-Tran III***, has a similar architecture but uses a faster field bus for inter-module communication, namely 1 Mbit/s CAN, and instead of the older controllers now relies on a 32 bit *HD64F7047* as the main-processor and a *HD64F3687* plus 2 *HD64F3694* (all from RENESAS CORP., Japan) as sub-processors.

The last example given here is the architecture of the ***MICHE*** system, developed at MIT, Boston. This lattice type reconfigurable robot incorporates 2 processors. The main or primary processor (PHILIPS) has an ARM processing core, whereas the secondary microcontroller is an 8 bit processor from CYPRESS MICROSYSTEMS. Again, I^{2}C is used for the intra-module communication. Within an organism, IR based communications with a speed of 9600 bits/s has been implemented (Gilpin *et al.*, 2008).

2.2.2 General Hardware Architecture in SYMBRION/REPLICATOR

In this section, the electronic hardware and architecture of single SYMBRION/REPLICATOR robot modules (the first prototype) is described in more detail as another example of self-reconfigurable robots (see Fig. 2.9). In order to better understand the design considerations, the hardware requirements of the SYMBRION/REPLICATOR robots – many of which may be generally valid for autonomous reconfigurable robots – are given below:

1. Every robot module be capable of moving freely and autonomously in a 2-D plane.

[1] I^{2}C™ (Inter Integrated Circuit) is a registered trademark by the PHILIPS ELECTRONICS N.V. CORP., Netherlands.

[2] IrDA™ is a registered trademark of the INFRARED DATA ASSOCIATION CORP., USA.

[3] LON™ (Local Operating Network) is a registered trademark by the ECHELON CORP., USA.

2. Every robot module have a certain intelligence based on a pre-loaded operating program.
3. The operating program be loadable through its communication module.
4. Every robot module be able to determine its remaining energy level.
5. Every robot module be capable of recharging its battery from a dedicated charge station.
6. If fellow robot modules are short of energy, a module should be able to share its energy resource through a power bus.
7. Every robot must be able to communicate with other robots disregarding if it is a separated individual or if it is integrated in an organism.
8. Every robot be able to mechanically connect itself with other modules through an on-board docking system to form an organism.
9. The docking system should provide not only a stable mechanical connection, but also an electric connection for energy sharing and wired communication.
10. Every robot should have a sensing system for accurate docking.
11. The organism be capable of moving freely and autonomously in a 3-D space.
12. For orientation, self-awareness, and cognition, every robot should carry sensors, such as accelerometers, infrared detectors for ranging, microphones or cameras. However, not every robot must necessarily wear the same set of sensors.

Since in SYMBRION advanced control and evolutionary algorithms, such as on-board genetic evolving, etc. needed to be implemented, here, one major design criterion was the calculation and processing speed. On the other hand, REPLICATOR required a high number of different sensors since the swarm's objective was to form a highly dynamic sensor network for vast applications, like surveillance, exploration, etc. As shown in Fig. 2.9, each module hence carries a number of processors/microcontrollers. However the major control of each robot is performed by the "Core Processor", an *LM3S8970 Cortex*[4] microcontroller from LUMINARY MICRO INC. The main purpose of it is to pre-process raw sensor data, to run higher level algorithms such as an artificial immune system (AIS) or artificial homeostatic hormone system (AHHS) as described in Sects. 4.4 and 4.2, to calculate the module's position, to pass this information to actuators, etc. In order to support this processor, a shadow processor (*Blackfin*[5] *ADSP-BF537E* from ANALOG DEVICES) is included that mainly takes over computationally intensive processing tasks, i.e. of the images taken from the 4 on-board cameras (s. Sect. 2.2.4). Due to its high power consumption, the intention is to operate this processor unit only if required. For example, if image processing has to be used to recognize the environment or if the organism size (i.e. number of docked modules) reaches a certain limit so that locomotion tasks require a lot more computational resources.

A dedicated microcontroller (*ATmega1280* from ATMEL INC.) is responsible for A/D-conversion and further processing of analogue sensor signals like microphones, IR-based distance sensors, etc. Since at least 1 brushless motor, whose control occupies many processing resources, is on board a robot module 2 additional Cortex

[4] *Cortex*™ is a registered trademark of ARM LTD. CORP., United Kingdom.
[5] *Blackfin*™ is a registered trademark of ANALOG DEVICES INC., USA.

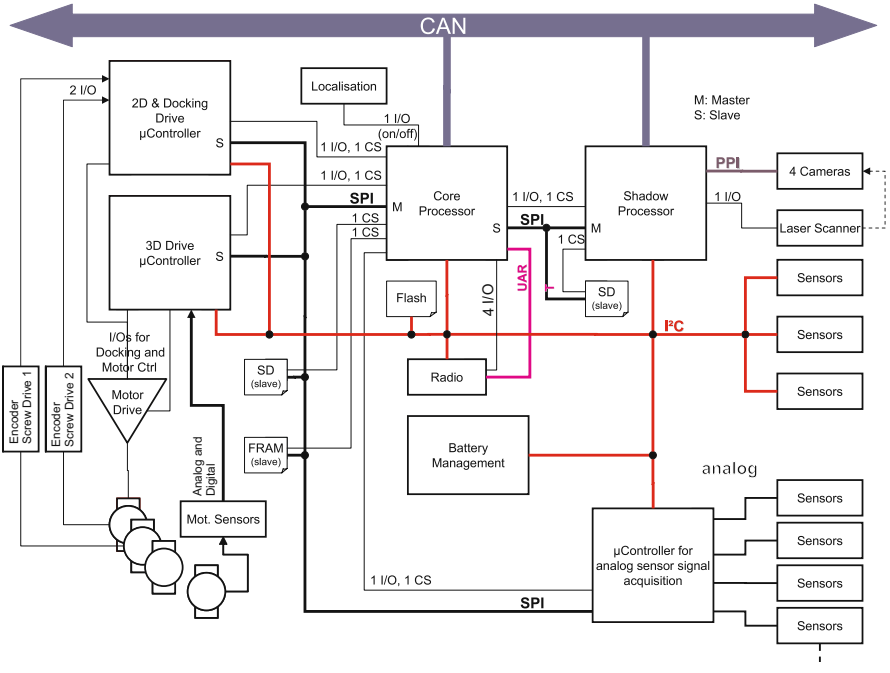


Fig. 2.9 Electronic architecture of the SYMBRION/REPLICATOR robotic modules.

controllers (*LM3S8962*) have been integrated, dedicated to all major actuation and locomotion tasks. Furthermore, the robots possess a UWB-based localisation unit as described in Sect. 2.2.6, a *ZigBee*™ radio communication module, a battery management module, Flash and SD memory, a LASER ranging module, and other sensors.

In order to be able to compare the capabilities of the various microcontrollers mentioned in this chapter (not only those that are used by SYMBRION/REPLICATOR), a brief overview is given in Table 2.2.

The main intra-module communication channel in SYMBRION/REPLICATOR is an I^2C-bus. Most of the sensors are attached to this bus. Since I^2C is a multi-master bus, it is straight-forward to connect all available processors to this bus. Depending on the implementation, I^2C permits transfer rates of up to 1 Mbits/s. However, the net data rate is lower since each device must first be addressed through address coding which adds information to be transferred. In addition, some sensors are only able to handle a gross rate of transmission of only up to 400 kbits/s. Two separate SPI-buses are therefore used additionally for faster inter processor communications. This is a true master-slave bus that uses CS-lines for addressing and interrupt requests for slave communication initiations, so that very little overhead is generated. The shadow processor is master of SPI-bus 1, whereas the core processor is master of SPI-bus 2.

Table 2.2 Basic features of the given microcontrollers.

Controller	Architecture	Clock	Power Consumption	SRAM	Flash	EEPROM
ATMega8	8 bit RISC	16 MHz	3 V, 3.6 mA at 4 MHz	1 K	8 K	512 Bytes
ATMega16	8 bit RISC	16 MHz	5 V, 12 mA at 8 MHz	1 K	16 K	512 Bytes
ATMega128	8 bit RISC	16 MHz	5 V, 17 mA at 8 MHz	4 K	128 K	4K
ATMega1280	8 bit RISC	16 MHz	5 V, 10 mA at 8 MHz	8 K	128 K	4K
LPC2148	16/32 ARM7	60 MHz	3.3 V, 40 mA at 60 MHz	42 K	512 K	-
PIC16F873	8 bit RISC	20 MHz	3.3 V, 0.6 mA at 4 MHz	192 B	4 K	128 Bytes
PIC16F877	8 bit RISC	20 MHz	3.3 V, 0.6 mA at 4 MHz	368 B	8 K	256 Bytes
HD64F7047	32 bit RISC	50 MHz	5 V, 200 mA at 40 MHz	12 K	256 K	-
HD64F3687	16 bit RISC	20 MHz	5 V, 21 mA at 20 MHz	4 K	56 K	-
ADSP-BF537	32 bit RISC	600 MHz	1.3V, 227mA at 600MHz	32 K[a]	48 K[b]	-
LM3S8970	32 bit RISC	50 MHz	2.5 V, 52 mA, at 50 MHz	64 K	256 K	-
LM3S8962	32 bit RISC	50 MHz	2.5 V, 52 mA, at 50 MHz	64 K	256 K	-

[a] L1 Data SRAM, [b] L1 Instruction SRAM; (ATMEL, 2009c; ATMEL, 2009b; ATMEL, 2009a; ATMEL, 2007; NXP, 2008; Microchip, 2001; Renesas, 2004; Renesas, 2005; Analog Devices, 2009; Luminary Micro, 2008; Luminary Micro, 2009)

Since the multiple modules within an organism need to cooperate and distributed computing is envisaged for the aggregated system, a reliable communication bus is required that is fast enough to pass on the data from module to module with acceptable delay. For SYMBRION/REPLICATOR, distributed world modelling is strived for, making strong use of the vision system. This implies that large amounts of data will most probably need to be transmitted across the bus. In the first prototype, CAN has been implemented since it is a rugged multi-master field bus, which is most favourable for this kind of application. However, due to the speed restriction of 1 Mbit/s this may turn out to be a bottle neck for a larger organism. Alternatively, *FlexRay*[6], a successor to CAN, provides a tenfold of speed. Since it is a relatively new standard, only very little hardware is currently available. Therefore, also 100 Base-T Ethernet using a network switch to handle communications from the 4 docking ports is under consideration.

2.2.3 General Sensor Capabilities

Following the approach from the previous section, we consider now the general sensor capabilities of the platform. For the application of evolutionary approaches as well as for sensor network applications, the platform should provide a measurement of environmental values, in particular, how robots do fit to the environment. The local fitness measurement for collective behaviour represents a very challenging task, therefore a serious attention during the design of SYMBRION/REPLICATOR platform was paid to this issue. From a conceptual viewpoint, the following four

[6] *FlexRay*™ is a registered trademark by the DAIMLERCHRYSLER AG CORP., Germany.

Table 2.3 Several examples of on-board fitness measurement.

Type/Name	Can be used for
Approximation of a global state	
Exploration of a global map	Goodness of behavior
Global coverage	Goodness of behavior
Position/Orientation in 3D space	Success of kinematic transformations
Local environment	
Gradient of light	Environmental feedback
Gradient of temperature	Environmental feedback
Number of neighbors	Feedback of collaborative strategies
Number of collisions	Goodness of behavior
Distances to objects	Goodness of behavior
Specific signals in environment	General feedback
Explored area	Goodness of behavior
Robot-Robot	
Internal states of another robot	Feedback of collaborative strategies
Number of received "eggs"	Feedback of collaborative strategies
Trophallaxis exchange	Feedback of collaborative strategies
Internal	
Energy Level	Individual fitness/activities
Distribution of energy	Individual fitness
Number of internal failures	Individual fitness

ways are available to measure the fitness: approximation of a global state by local sensors, perception of local environment by on-board sensors, different measurements during robot-robot interaction, and finally, measurements of internal states. These factors are summarized in Table 2.3.

1. Approximation of a global state by local sensors. For an application of evolutionary strategies the most appropriate feedback may be provided when knowing a global state of the environment, including internal states of other robots. However, such information is not available for individual robots due to practical reasons. Nevertheless, the global state can be approximated when using the world model and several sensor-fusion approaches, see Sects. 3.1 and 3.2. Examples of global states are map-related values, such as explored/unexplored area, coverage of some territory, position of robots in 3D space. The platform includes several sensors, such as a localization system or laser rangers, for these purposes.

2. Sensing a local environment. Perception of local environment by on-board sensors is the primary way of receiving information about the environment for both evolving and sensor network applications. The overview of integrated, or considered for integration, sensors is given in Table 2.4.

Table 2.4 Integrated, or considered for integration, on-board sensors.

Sensor	Name	Interface
Environmental		
Light	ADPS9002	analog
Air Pressure	SCP1000	I2C
Directional Sound	SPM0208HD5	analog
Humidity/Temper.	SHT15	I2C
IR-reflective	TCRT1000	analog
Imaging Sensor	OV7660FSL	PPI
Laser (in the Range Finder)	LS-1-650	digital
RFID sensor	Lux	no
Sonar sensor	SRF08(or 10)	I2C
Laser Range Finder	URG-04LX	RS232/USB
Detecting motion in environment	AMN34111	analog
Hall effect (magnetic)	US4881EUA	analog
Color Sensor	TCS230	digital
Capacitive	MT0.1N-NR	digital
Locomotion		
3D Acceleration	LIS3L02AL	I2C
WTL laser mouse	ADNS-7530	SPI
3D Localization	Ubisense	digital
Orientation-Sensor	SFH 7710	SPI
IR-docking sensor	IR-based	analog
Force measurement sensor	K100N/RB-Int-01	analog
Joint angle sensor	2SA-10-LPCC	analog
Compass	HMC5843	digital
Internal, Indirect Sensors		
Voltage, Current	BQ77PL900DL	SMBus
Bus Load Sensor	no	software
Center of mass	no	software
Energy-docking sen.	no	software

3. Information provided by a robot-robot interaction and communication. Robot-robot interaction is a very important source of fitness measurement. The corresponding sensors are the force measurement sensors, joint angle, compass or 3D accelerations. Robot-robot communication plays also an important role here, which allows fusing local information from different robots. This is related not only to environmental values, but also to internal states of robots.

4. Internal states of robot organisms. There are different internal sources of information: energy-based, mechanical, load on buses, number of internal failures, CPU/Memory usage and other. The energy-based values, discussed in Sec. 2.3, are very useful for many purposes, e.g. in estimation of the most efficient structure of organisms. Generally, the number of internal sensors, most of them are virtual sensors, can be very high.

To give a reader an impression about sensing capabilities of the platform, we collect in Table 2.4 a brief overview of on-board sensors. In the following we consider the development of specific vision and IR-based sensors in Sect. 2.2.4, a smart laser sensor in Sect. 2.2.5 and a localization system in Sect. 2.2.6 in more detail. This consideration is finalized by outlining the integration issues in Sect. 2.2.7.

2.2.4 Vision and IR-Based Perception

In the previous section, we considered different off-the-shelf sensors, which will be integrated on the platform. This section describes a development of specific vision- and IR-based perception, where we fist concentrate on imaging system. In a robot organism, the robot cells perform a lot of tasks ranging from locomotion, 3d actuation, communication, sensing, power management and control, thus, performing image processing creates a burden on the cells processing capabilities due to the large amount of images.

Parallel and distributed image processing provides a solution for the large amount of information provided by cameras. Having four cameras on every robot cell, a robot organism of nine cells, as an example, has about 20 accessible cameras depending on the cell formation of the organism. This makes parallel and distributive image processing a necessity in such a multi-camera architecture (Aghajan & Cavallaro, 2009). Fig. 2.10 shows an example of a robot organism consisting nine robot cell formation and the accessible and non-accessible cameras. Also, Fig. 2.11 shows an example of four consecutive images grabbed from the vision prototype.

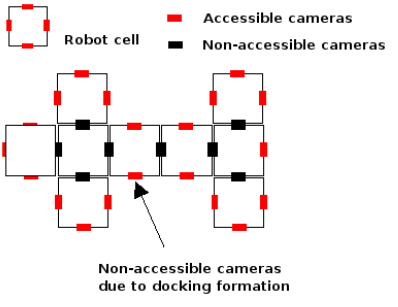


Fig. 2.10 An example of a robot organism.

Parallel and distributive image processing relies on the Communication Backbone (CBB) for Load Distribution. Fig. 2.12(a) shows a diagram with an example of a Communication Backbone in a robot organism. The communication medium distributes the processing work load all over the whole system. For example, robot cells busy with 3d locomotion can grab an image and transfer it over the communication medium to another robot cell for processing. Fig. 2.13 shows an example of work load distribution in a robot organism.

Under heavy image processing, the robot cells of the organism will act as a pipeline, where every robot cell will take an image processing subtask thus speeding processing and increasing its image processing capabilities. To be capable of doing immense image processing the robot organism relies heavily on its communication medium backbone. The result of every finished subtask will be put on the communication medium for the next cell in the pipeline to process. The end result of the last cell in the pipeline will be sent to the master robot cell running the scenario which

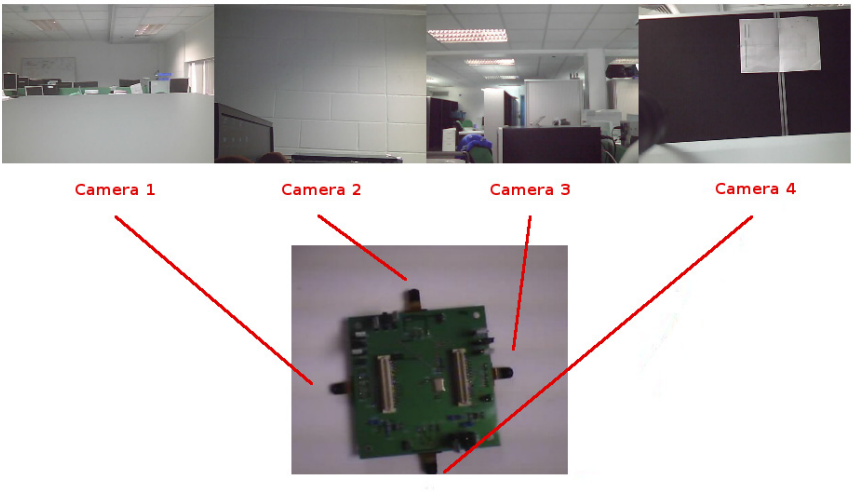
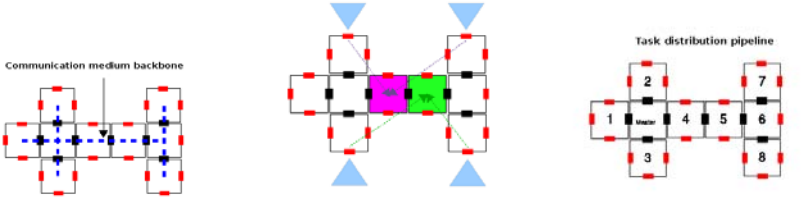


Fig. 2.11 An example of four consecutive images grabbed from the robot vision prototype.



(a) (b) (c)

Fig. 2.12 **(a)** Communication medium Backbone of the robot organism; ; **(b)** Parallelism of stereo vision task on two different robot cells having two different sets of stereo images; **(c)** An example of an image processing task distribution pipeline in a robot organism showing the master robot cell which initiated the task and the robot cell arrange with respect to their position in the pipeline.

initiated the image processing task. Fig. 2.12(c) shows an example of a task pipeline in a robot organism.

Initiating parallel and distributive image processing tasks relies on the Master Robot Cell (MRC). The master robot cell is the cell which initiates the request for distributing an image processing task. The master cell checks the processing work load of every cell in the organism and identifies the cells needed for pipelining. Then, it arranges the image processing subtasks for every cell creating the pipeline. The same procedure is followed for the master robot cell to initiate two parallel tasks that each run on a different cell performing the same task but with different set of image data.

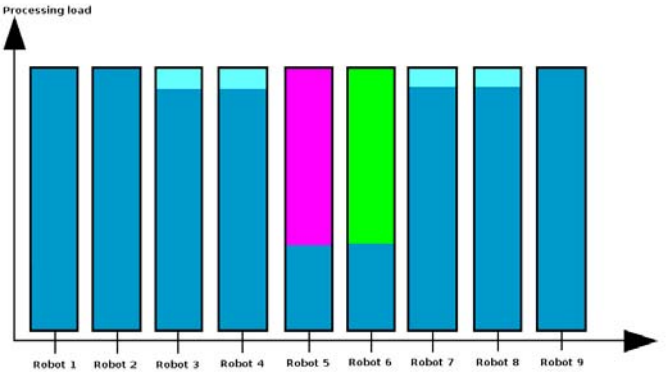


Fig. 2.13 Diagram showing the processing workload of every robot cell in the organism and the workload assigned to robot cells with low processing workload.

Any robot cell in the organism is capable of grabbing images from any camera in any robot cell in the organism and it also capable of retrieving any image and image processed information in any memory in any robot cell in the organism. This distributed vision architecture provides the robot organism with vision rich information, flexibility and complete control for vision algorithms. This feature relies on to aspects the communication medium backbone and on an intelligent camera and memory selection algorithm. Fig. 2.14 shows an example of a consecutive image grabbing from every accessible camera in the robot organism showing the necessity of camera selection control in a parallel and distributed vision system.

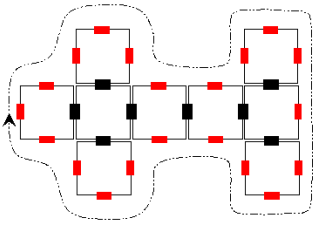


Fig. 2.14 An example of a consecutive image grabbing from every accessible camera in the robot organism showing the necessity of control in a parallel and distributed vision system.

The selection algorithm relies on a continuously updated database of all the robot cells and their corresponding accessible cameras in the organism. Thus, it needs updated information of cell formation of the robot organism to know which cameras are not accessible due to docking formation. The other aspect that the selection algorithm relies on is complete control of any camera on the robot organism, thus, providing an accessible medium for any robot cell in grabbing images from any camera in the organism.

IR-based Guidance for Docking and Obstacle Detection

Another developed system of the platform is the IR-based perception. One of the key requirements for the robots is the capability for autonomous morphogenesis:

(1) the robots can move freely without colliding with each other and (2) can dock to (and undock from) specific locations of the organism as required. The general idea for the docking approach is illustrated in Fig. 2.15: once one robot (A) in the swarm decides to initialise the docking process it will broadcast some signal to attract other robots.

Another robot (B) within range can detect the signals using an array of sensors and move towards the signalling robot along the detected direction. Thus docking can be divided into three stages according to the range between A and B:

1) recruitment – B detects A's 'docking' signal at long range but it can only deliver approximate direction information;
2) docking alignment – B executes precise alignment at short range;
3) docking ready – A and B are close enough that the physical docking mechanism can take effect and lock cells together.

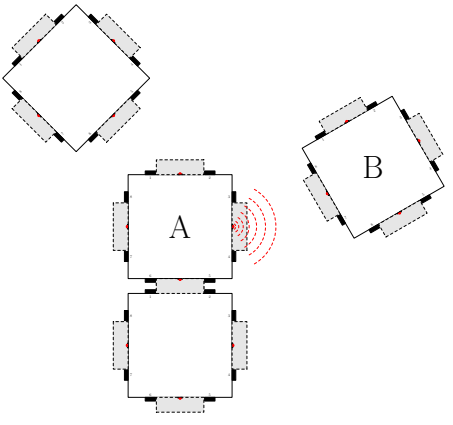


Fig. 2.15 Scenario for autonomous docking, B docks to A.

To accomplish such a procedure, either sound or light signals can be used. However, considering the power consumption, the physical dimensions, the operational range and also the commercial availability of the sensors, we choose to use only Infra-Red signals in our implementation. Three different sensor units with different operational range are developed for these purposes. The functionality of these sensor units can be categorised to be obstacle detection (proximity sensor), beacon detection (docking sensor) and local communication, according to their operational range, which also correspond to stages 3, 2, 1 respectively. As each robot has a cubic shape (size: $80 \times 80 \times 80$mm) and docking is allowed on four sides, multiple identical sensors will be distributed around the robot, as shown in Fig. 2.16. Table 2.5 lists the essential components used in our design against the proposed functions.

Table 2.5 The usage of sensors against the functionality.

Functions	IR sensor[1]		LED[2]	IR receiver[3]
	emitter	receiver		
Proximity	✓	✓		
Docking		✓	✓	
Communication			✓	✓

[1] *TCRT1010*; [2] *TSML1020*; [3] *TSOP36236*.

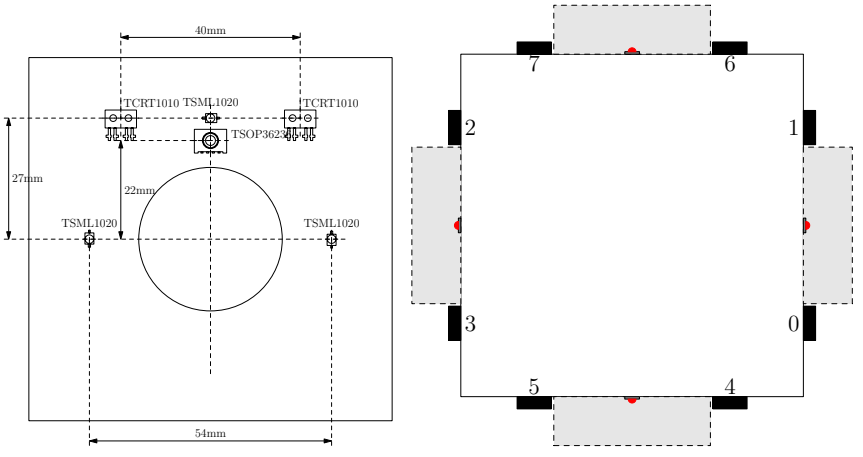


Fig. 2.16 Placement of the sensors: **(a)** Side view; (b) Top view.

Here, obstacle detection plays two roles during the docking approach process: (1) detecting the obstacles when approaching the signalling robots and (2) detecting the distance to the signalling robots for docking alignment (with the beacon detection sensors introduced in the next section). As illustrated in Fig. 2.16(b), 8 IR sensors (marked with dark rectangles) are placed on the four side PCB boards, two on each side.

Successful autonomous docking between two robots requires reasonable accuracy of alignment prior to docking. The idea to achieve the alignment is somewhat like beacon detection: one robot flashes an IR LED at a fixed frequency - placed right above the docking unit - acting as a beacon, while the other robot uses its 8 IR sensors (the same ones used for obstacle detection) to detect the signals. To deal with the situation that the robot may turn 90° in both directions, on each side PCB, two extra LEDs are placed on both the left and right sides close to the docking units, as shown in Fig. 2.16(a).

As the obstacle detection and beacon detection approach rely on the analogue output of the IR sensors, they work only in very limited ranges. In order to achieve autonomous self-assembly in a large swarm of robots, sensors with longer operational range and local communication mechanism are required to recruit more robots for the organism transform process. Four different IR communication channels are implemented for each robot, one channel on each side. These four channels can work individually. By default, all four channels are in 'listening' mode. Whenever one robot is broadcasting messages, another robot within range will receive the message with one or two adjacent channels, which provides the robot with rough directional information about where the signalling robot is.

2.2.5 *Triangulation Laser Range Sensor for Obstacle Detection and Interpretation of Basic Geometric Features*

The further developed sensing system is the triangulation-based laser range sensor. In order to perform far-range exploration of the environment, safe navigation (e.g. obstacle avoidance), localization and mapping tasks in uncertain environment, usually several range measurement sensors are embedded on mobile robots, e.g. ultrasonic sensors, light (e.g.,infra red, IR) sensor and specifically laser sensor. Among these sensors, laser-based range sensors and scanners can offer highly focused structured light, this can be used for a more refined and computationally-simpler understanding of the unknown environment by elaborating the structured-light pattern as acquired by an integrated camera. In addition, structured light allows the robot to make relatively accurate measurements of geometric features of the surrounding objects, contributing to the decision-making process of the robot in order to accomplish many tasks, or more simply for safer navigation and exploration. There are two basic working principles exploiting lasers for measuring and scanning applications:

1. Time-of-flight (TOF) or phase-shift based system: basically exploiting the known light speed as parameter for distance calculations of (non-modulated or modulated) light propagation;
2. Triangulation-based system: In the system, the laser beam emitted by a laser diode is projected onto the surface of an object, and the reflected beam is gathered back by an image sensor (CCD or CMOS). Position of reflected beam in image plane, optical center of camera, source of laser beam and laser point on object surface constitute basic geometrical setup, obstacle distance from camera to object can be computed from this geometrical relation.

TOF laser sensors offer measurement range up to tenths of meters, but they need a quite accurate and cumbersome electronics which is not suitable to be integrated in miniaturized mobile robots. Cost issues are also relevant for large multi-robot systems if the integration of such a device is desired for each individual robot.

Triangulation laser sensors require much simpler electronics and can be integrated in small volumes, though offering much shorter distance range capabilities. However, in particular for large multi-robot systems, and even more important if miniaturized robots are targeted, the capability of a local sensing is perhaps the most important, while far-range information acquired by one of modules can be shared in a multi-agent or swarm system. Considering the target of equipping each of the many planned robots with this system, the sensor cost turns to be the most important factor. The simple hardware allows small size, lightweight and low power consumption.

Taking advantage of the presence of many robotic modules and their capability to dock to an organism, the first idea can focus on the possibility to have a cooperative laser scanner, by exploiting the capability of the assembled organism to scan an object with an “arm” equipped with a laser-line generator, while detecting the image with a camera mounted on another physically displaced “arm”. In such a case a known laser-camera distance and declination of laser beam over optical axis of

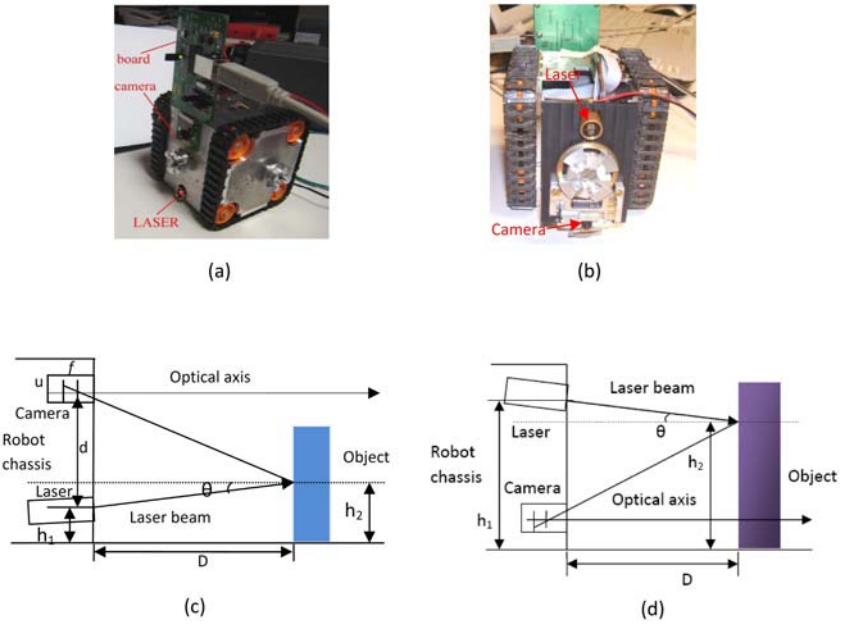


Fig. 2.17 Configurations of triangulation laser sensor on robot: **(a)** Laser and camera on front side of robot; **(b)** Laser and camera on rear side of robot;**(c)** Sketch of configuration for laser tilting up small angle on front side of robot;**(d)** Sketch of configuration for laser tilting down small angle on rear side of robot.

camera had to be precisely determined to obtain the basic geometry for distance calculation by analysis of kinematic chain that connects two "arms" mentioned above. In this way it would have possibility to explore small objects by reconstructing three-dimensional shape of them using a 3D model building algorithm.

However, the possibility to have a reliable scanner by means of cooperative "arms" appeared quite challenging, in particular regarding the precision that could be obtained, heavily depending on the accuracy of kinematic chain of the organism. In addition, the system could be exploited by the organism only in particular circumstances. Consequently, it would more preferable to try to develop a miniaturized laser scanner onboard each robot, in order to exploit the laser scanning technique not only at the organism level, but also in the single robotic module, thus giving much more autonomy and capability to the individual robots themselves.

In this way, the triangulation laser sensor having both the laser-line generator and the camera onboard each robot was chosen for the proposed system. Since the choice of the camera and the distance between the laser and the camera can be determined based on the other sensing requirements and the mechanical design, the relative position of the laser and the camera and the tilting angle of the laser over the optical axis of camera are main parameters to determine the robot-object distance, resolution and minimum height of the detectable objects. The possible configurations

investigated to determine the optimized parameters are shown in Fig. 2.17. For the configuration shown as Fig. 2.17(a), the range sensor can achieve far range and good resolution, but short object and hole on floor will not be detected; For the configuration shown as Fig. 2.17(b), short object and hole on floor can be detected, but the detectable range is limited. Both configurations can achieve 2D static scanning, while the later configuration can achieve 3D active scanning through controlling movement of bending part.

The distance from camera to object D under the condition of 2D static scanning can be described:

$$D = \frac{fd}{u + ftan\theta}, \tag{2.1}$$

where f is focal length of camera, u is the distance from point of reflected laser beam on image plane to the middle of image plane, d is the laser-camera distance, θ is the tilting angle of the laser over the optical axis of camera. The positive value of the θ means that the laser tilts up and the negative value means that the laser tilts down.

The geometric features of surrounding environment(i.e. hole, gateway and corridor) can be understood after laser lines are extracted from background using image processing algorithm. The output data from triangulation laser sensor such as camera-object distance D, hole width, gate width, object height, and orientation angle of object can be used for motor control of robot, thus robot will be navigated effectively in an unstructured environment.

2.2.6 *Powerful Wireless Communication and 3D Real Time Localisation Systems*

This section describes two subsystems - wireless communication and 3D localisation, which are developed in a synergy. The powerful wireless communication system and active localization of robotic modules is capable of ensuring data transfer both within and outside of the robotic system. The system design comprises both hardware, based on state-of-the-art Cortex Chips and IEEE 802.15.4/ZigBee communication standard, and control software for those hardware components.

The powerful wireless communication system is built on a hardware and software platform of ZigBee chips or modules, thanks to which all the standard advantages can be exploited. The communication system is designed to exchange data from a range of sensors, and to monitor and control the robotic network. The plan is for hundreds of robots to co-operate within the network.

As a main software component, a Communication application interface (CAPI) has been created. CAPI ensures:

a) sending and receipt of general messages,
b) receipt of co-ordinates,
c) sending of notification and alerts,
d) switching off of radio module and activation of power saving mode, and
e) request for messages passed off in saving mode.

Every single radio module has its own UART driver implemented. CAPI defines data structures, alert message codes, error messages and function prototypes. CAPI functionality and robustness is tested within a network of about fifteen modules (which simulate real robots), in which network communication is charged by a large amount of data transfer. One of the main objectives of the tests is to examine the robustness and power budget in mesh topology. PC application software is created to drive radio modules and to simulate data.

The identification system concept starts out from two-level identification logic - passive and active. The passive level is implemented by powerless RFID tag implementation, the active level is implemented by unique WPAN addressing. A passive identifier serves for IdM (Identity Management) implementation. IdM, formerly established for a "living organism world", will ensure follow up of the entire life-cycle of units in a "robotic world". IdM will process such events as robot introduction into the system, monitoring of its developing features, robot character configuration (or skills) , activity performance, removal from the system and other events within the life-cycle.

The active localization system (ALS) has been developed through a synergy of two systems in project - wireless communication and 3D real time localisation. The active localization system came into being within the framework of this project and is useful for position detection with an accuracy of up to 15 cm within an area of up to hundred thousands square meters. The main benefit of ALS is enhancement of decision-making processes as a part of the artificial intelligence of the robotic system. In principle, every entity will be aware of its own position in real time (so-called spatial awareness), as well as the positions of other entities. Data provided by hardware components of ALS is processed and visualized on-line using software utilities. The essential advantage of the active localization is that allows an autonomous effective and fully real-time mutual co-operation of the robots. Robots do not need any commands from the upper system. Robots themselves are aware "where who is" and "what is intended to be done by whom". Robots would be fully dependent on the system decision making.

Communication System

In the SYMBRION/REPLICATOR we perceive the robot primarily as an autonomous entity, but also as an entity which is able to co-operate in a swarm. Similarly to living organism, when the swarm executes a common task, the activity of every unit, given by a role within the task, is different and highly dynamic. Connection via wires is inoperable or absurd in most of these cases.

ZigBee standard ensures various ways of building a device network within a so-called Personal area network (PAN). Profitable advantages such as - a) power saving, b) ability of huge amount of units to co-exist within the network, c) multi-hop and ad-hoc routing ensuring communication within a wide range – bring the first possible answer. The second answer we could look for is in the intention for which the standard has been established – state-of-the-art technology for an efficient approach towards automation, regulation, energy management, sensor network management,

to make communication in this field as easy as possible. Taking into account the fact the projects intend to create the robot as a carrier of variable sensors which will gather an amount of environmental data to provide to the industrial domain, the standard appears to be a good choice in order to make the robotic system industrially usable and interoperable. The standard also complies with swarm demands in terms of short-range communication. The operational radius of the swarm is expected to be around hundreds of square meters.

Similarly to the human world, the robotic world needs to communicate free of useless and redundant information flow and to use clear and unique commands. Furthermore, robots can be taught to increase their efficiency by using message-coding, which also helps to control the possible congestion of communication channels. Within the standard terminology, every robot in the network (swarm) is either an End Device or a Router (depending on the functionality requested) and the entire swarm controlled by the so-called Co-ordinator. In our robotic terminology the Co-ordinator is called the Base Station. The Base Station has two basic functions - a) to control the swarm, b) to mediate communication with other systems, e.g. the Localization system. From the perspective of the communication system concept, the efficiency is also affected by the implementation process. Management of the communication has to handle the data flows on several levels of software modules as well as hardware interfaces. Communication Management has to decide when it is better to communicate by wire, light, sound or by air, taking into account amount of data and the destination. Wireless communication is implemented into the Communication Manager through an application interface to abstract from the protocol implementation details and to make use of the wireless channel as simple as possible for programmers.

The communication principles, with respect to who is talking with who and for what reason, are designed so as to approximate the behaviour of living organisms. Even an organism in a crowd knows to address a message only to one or to all, as well as which it knows what level of sound to use to address only close neighbours, or what message content or language to select in order to be understood by the targeted organisms. According to the software design, the robot uses a set of system- and function- messages in order to achieve the objective of communication. In order to set the communication principle in operation, the implementation of various software functionalities under a robotic computation system, is required. Fig. 2.18

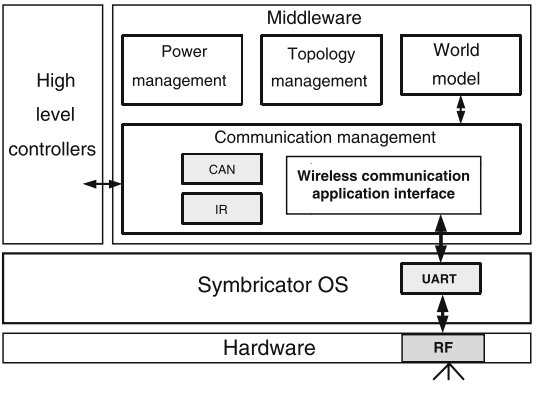


Fig. 2.18 Wireless communication application interface within SW architecture.

shows how the wireless communication application interface is integrated into the robotic computation system.

One of the basic assumptions for efficient communication is that every subject has to have unique identifiers. The reasons for identifying artificial subjects is similar to why this is obvious among living organisms - the entity when born obtains an ID, which will be linked to every activity performed by the entity during its life cycle until it dies. For this fundamental identification we can use a similar concept as that developed for the human-world called Identity Management (IdM), using RFID means functioning powerless (even non-powered entity has to be identified). Besides RFID means we intend to utilize two other identifiers for the advanced identification system - unique number of the RF module and unique number of the localization Tag. All identifiers are merged in the Base Station, they are used for message addressing within the wireless communication system.

3D Real Time localisation and Robotic Spatial Awareness

The 3D real time localisation is a very important topic as it rules the reaction of the robot swarm towards its changing position within the surrounding environment. Localisation techniques are mainly based on distance estimation, angle measurement, neighbourhood proximity or hop-count methods. All these schemes rely on physical signals and require their own parameters including hardware requirements, scalability aspects, performance metrics and constraints.

A number of different location positioning systems have been proposed during the last two decades. For instance, the active RFID and IR-based systems (Want *et al.*, 1992) have inherently limited indoor coverage. The ToF-based systems require an accurate synchronisation of all the nodes and any timing error in clocks of the transceivers induces ranging errors. Localisation methods based on RSSI (received signal strength identifier) have been investigated in (Lorincz & Welsh, 2005; Alippi & Vanini, 2006). However, all the RF based localisation and tracking systems require a prior calibration or an indoor propagation model (Jemai & Kuerner, 2008; Jemai *et al.*, 2008), which is not simple. Even if the model is well calibrated, the localisation errors due to unpredicted multi-path signal fading and shadowing could not be avoided (Jemai *et al.*, 2009; Jemai & Kürner, 2009). However, techniques based on time-difference of arrival (TDoA), though requiring two types of transceivers, provide more accuracy and robustness against multipath and signal fading (Gustafsson & Gunnarsson, 2003).

These techniques have been applied in several recent localisation systems. For instance, Massachusetts Institute of Technologys Cricket system (Gustafsson & Gunnarsson, 2003) is an indoor localisation system, which uses TDoA between a radio and an ultrasonic signal for distance ranging to compute a trilateration from pre-positioned nodes. The position is computed based on the distance estimates from Cricket and Bayesian filtering (Priyantha *et al.*, 2000). In (Ansari *et al.*, 2007), the localisation and tracking framework makes use of a non-linear Bayesian filtering scheme (Particle filters) to handle non-linear motion even in the presence of non-Gaussian measurement noise.

The Active Badge infrared-based (Want *et al.*, 1992) localisation system developed by AT&T beginning of the 90^{th}, one of the first developments, is based on localising people (roughly with a room accuracy) moving within a building by single infra-red receivers placed in different rooms. RADAR (Bahl & Padmanabhan, 2000) uses the signal strength and signal-to-noise-ratio of wireless LAN for indoor position sensing. In practice, clock synchronisation of all involved transceivers is often very difficult. Therefore, time difference-of-arrival (TDoA) techniques are required. The same group who developed the Active Badge system (Want *et al.*, 1992) proposed later the Active Bat system (Cox, 1991). It relies on Ultrasound-based time-of-flight lateration. A bat, which is carried around by a person, sends an ultrasound chirp to a grid of ceiling mounted receivers. Simultaneously, the receivers are synchronised and reset by a radio packet that is also transmitted by the bat.

Throughout the last decade, sensor-based real time position estimation of robots has been recognised as a key problem in mobile robotics. Therefore, there has been a tremendous scientific interest in algorithms for estimating a robot's location from sensor data (Borenstein *et al.*, 1996; Burgard *et al.*, 1996; Cox, 1991). In other several recent papers, the authors have used an infrared technology for robots localisation (Chae *et al.*, 2006). In (Espinace *et al.*, 2008) the structural information is used in conjunction with Monte Carlo Localisation strategy to estimate the robot positions. The authors in (Nuechtera & Hertzberga, 2005) presented a laser-based approach for tracking the pose of a high-speed mobile robot. Most recently, in (Eckert *et al.*, 2009), the authors provided a framework for time-of-flight based localisation systems relying on ultrasonic sensors for the localisation of quadrocopters flying robots.

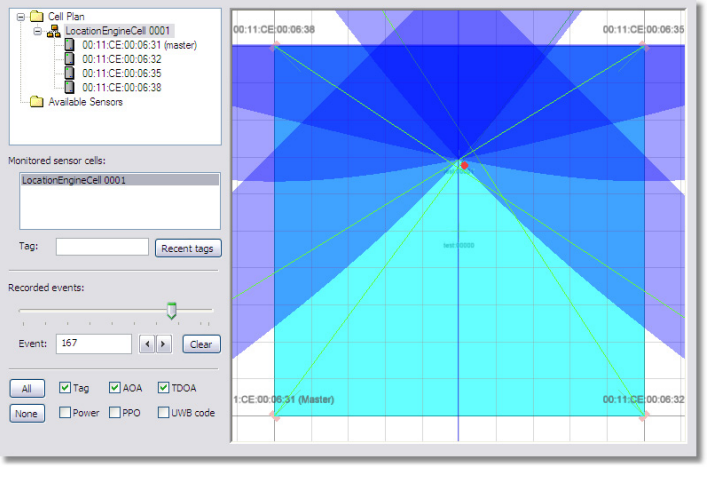


Fig. 2.19 Localisation principle with TDoA hyperboloids (blue) and AoA lines (green)

In this work, we propose to apply a novel robotic localisation system, characterised by its unique combination of the angle and distance information in order to deduce a very accurate position. This feature makes the system distinguished from all the previous related works described above. In order to realise a three dimensional real time localisation, each robotic module has been equipped with a miniaturised active transponder (tag) which transmits trains of very short ultra wideband pulses in the frequency band of 6.5-8 GHz. These signals are detected and processed by the sensor network and a location engine (running on a server station) connected together via the Ethernet network (Ward *et al.*, 1997; Harter *et al.*, 1999; Harter *et al.*, 2002).

The sensor network in composed of a master (which rules the bidirectional communication between the tags and sensors), a timing source (which provides the synchronisation clock to receive the ultrawideband pulses) and a number of slave sensors hearing the tag pulses and forwarding them to the master in real time. The sensor network is installed at fixed known positions in the surrounding localisation environment (e.g. at the corners of a room) forming a location cell. Having already measured reference coordinates, each sensor should have its angles calibrated with reference measurements at known positions within the tracking area. The location engine derives, by trilateration, potential positions of the localisation tag, situated on a hyprboloid. Furthermore, each fixed sensor has an antenna array which enables extracting the angle of arrival (AoA) information from the coming pulses with an accuracy around 1°. In conjunction with the measurements, a Calman tracking filter accepts or discards the measured sighting events depending on a priori known parameters such as speed, height of the object and the error standard deviation (horizontal and vertical). The intersection of the angle of arrival lines with the TDoA hyperboloids, see Fig. 2.19, provides the position of the tag.

This unique combination of AoA and TDoA solves the problem of TDoA pathological geometry, which also provides the ability to gain locations from fewer sensors. Therefore, with small sensor density, this method delivers repeatedly a substantially more robust and reliable system than a TDoA only system. Furthermore, an additional advantage is the ability to provide a lower sensor density to cover areas where less precise location coverage is acceptable. This flexibility allows optimal price/performance tradeoffs to be made; offering the

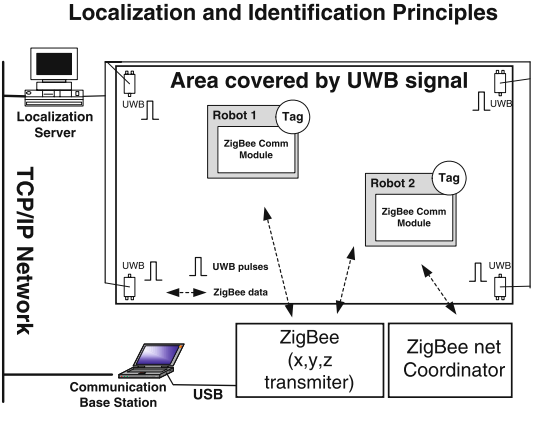


Fig. 2.20 Localization principle of the robot swarm and artificial organism.

ability to cost-effectively configure and utilise a single homogeneous system across the entire site.

Thus, the location engine enables an accurate real time tracking of the tags over wide areas. Depending on the number of tags present within a sensor cell and whether the robotic modules are moving or rather keeping static positions, the location engine performs an automatic dynamic update rate scheduling of localisation. Hence, robotic modules can be localised each 30ms up to each 2 minutes. Situated at a layer above the location engine, the location platform is for building robust sensor-driven location-aware systems. It performs a scalable persistent storage and publishing of real-time data and enables a user-definable 2D and 3D object and building visualisations. Moreover, it provides a real-time monitoring of user-definable spatial relationships between objects, robots, persons and assets.

In order for each robotic module to be aware of its position and the position of other modules in its surrounding, the location platform is connected over the Ethernet network to the ZigBee coordinator, which rules the communication between multiple nodes (robotic modules). The location of each tag is transmitted via TCP/IP to the ZigBee coordinator connected to the same network as shown in Fig. 2.20. The latter forwards the localisation information to each corresponding robot in order to react autonomously and in intelligent way.

2.2.7 Integration Issues

This last section is devoted to the challenging issue of integration of all components on the robot platform. Designing the architecture of a robot platform becomes difficult because many decisions and compromises must be made. After all, mechanics, electronics, software, and power sources must operate smoothly together and need to be integrated into a geometry that complies with the original specifications of the module and the envisaged organisms. Size, weight and power consumption should be kept minimal to ensure reasonable operation times and allow for the key feature of self-reconfigurable robots: flexibility of morphology. If modules are too heavy to be lifted by fellow modules, only a very limited number of different morphologies can be acquired, and the organisms may render themselves useless for operation in 3D space. Furthermore, experience has shown that modules must be robust if one of the "Grand Challenges" that have been identified in robotics should ever be realized (Yim *et al.*, 2007; Murata & Kurokawa, 2007). Two specific "Grand Challenges", reflecting long-term self-sustaining and evolutionary self-development, are selected as benchmarks for SYMBRION/REPLICATOR projects.

By definition, the robot cells closely cooperate and come into close physical contact. When operating in 3D space and in particular when on-line learning algorithms are implemented organisms will most probably topple in one or the other occasion. Therefore, the hardware should be integrated in a way that it can cope with this kind of "accidents". Self-healing of organisms by reconfiguration and disposition of broken modules does indeed increase the robustness of the whole swarm. But, since robots cannot replicate like living beings, the probability of survival in a fixed

time period is greatly increased when the modules are rugged. Complex architectures require many interface lines within the robot module. If the structure in itself is moveable, then it is wise to keep the number of lines from one part of the module to the other as low as possible. The reason for this is simple: every line that goes across and is exposed to frequent bending increases the likelihood of a system failure. This is one explanation why communication buses with low line count (such as I^2C) are so popular in reconfigurable robots.

A few final words should be also devoted to general engineering problems. Robotic projects, aiming not only a development of technology but also a demonstration, face among scientific and technological challenges a series of engineering problems. In several situations, not proper engineering design can hinder a demonstration of excellent ideas and vice-versa, an excellent engineering work can make some ordinary ideas more attractive. In this context, a serious issue is related to the "pragmatism" of proposed solutions. It means that potentially interesting approaches are often skipped due to insufficient engineering capabilities. Normally, during the cycle research – pre-development – development, parallel approaches in the research phase are reduced to more pragmatic solutions during the developmental phase. Strong focusing on a low number of alternatives in research may lead to a loss of potentially interesting solutions. It seems that finding a compromise between pragmatism and parallelism in the pre-developmental phase is one of the essential points for the success of projects

2.3 Energy Autonomy and Energy Harvesting in Reconfigurable Swarm Robotics

Raja Humza Qadir, Oliver Scholz

Any creature in nature needs energy to survive and spends a lot of its life time to regain its consumed energy resources. Similarly, in robotics, energy is a key issue that significantly determines the level of a robot's autonomy: A robot that is wired to a socket can at maximum move within a circle that is dictated by the length of the cable. But, there is no general time restriction in doing so. On the other hand, a robot that carries its own energy source on-board can move any distance as long as its resources are not depleted. Here, there does exist an intrinsic time restriction. However, the operation time can be increased by fitting the robot with a feature that enables it to harvest energy from the environment. Good examples are the diverse solar powered robotic rovers that explore Mars.

In swarm robotics, various methodologies have been employed to increase the autonomy, i.e. operational time without human interference, of an artificial swarm. One such bio-inspired method is trophallaxis, where one robot "feeds" another one in order to increase the operational range of the whole swarm (see Sect. 2.3.3). When individual robot modules joint to form a multi-robot organism, the distribution of energy among the modules becomes exceedingly interesting. In the very moment when the modules dock to each other, it is very likely that the charge levels of the batteries differ. Furthermore, although the organism as a whole will achieve

a common goal, the individual tasks and with these the power demand will vary among the modules within the organism. Hence, if not smartly managed, operation time of an organism will be dictated by the weakest of its modules.

One step towards longer operation times of collective robot organisms is to distribute and share the energy that is available in the individual robots' batteries. However, additional problems may occur in this case. Depending on the particular hardware and the number of modules (cells) that form the organism, care has to be taken not to approach an overload condition. To avoid this, some kind of energy management and task planning is required that is able to estimate the required power levels and has information about the individual battery conditions or status.

2.3.1 Energy Autonomy

The term *autonomy* comes from Greek, that essentially means "independent of", having the "ability or freedom to determine one's own actions and steer its behaviour". In philosophy, it is defined as one's ability to self-govern, self-direct and self-rely.

In swarm robotics, the *energy autonomy* of an autonomous robotic module is its ability to regain its depleted energy from the environment or surrounding in the presence of several other competing modules. To be energetically autonomous a robot module has to fulfil the "self-sufficiency" criteria. The *self-sufficiency* of an artificial robotic system is its "ability to sustain itself in a viable state for a longer period of time" (McFarland & Spier, 1997). For that, an autonomous self-sufficient robotic system has to comply with the "basic cycle of work" also defined by McFarland (McFarland & Spier, 1997). The basic cycle of work defines the ability of an autonomous system to find fuel – and refuel. The applicability of such a cycle is mainly dependent on two factors: firstly, the environment in which the robots are deployed, and secondly, the level of autonomy that influences the robot's behaviour. The *level of autonomy* defines the control of an autonomous system over various system parameters that influence its response or behaviour in a dynamic environment.

> "**Autonomous systems**, are the systems that develop for themselves the laws and strategies according to which they regulate their behaviour: they are self-governing as well as self-regulating. They determine the paths they follow as well as steer along them." (Steels, 1995)

In a multi-robotic system that involves cooperation among the modules at multiple levels of a robotic swarm the degree of autonomy of each module is coupled with energy autonomy. The degree of autonomy also depends on the degrees of freedom in the system, which is usually determined by the number of resources an individual module has to manage or bring along (Spier & Mcfarland, 1986). To gain energy autonomy an autonomous module must be capable of firstly, monitor its dynamically changing health status – the on-board energy source parameters, e.g. the state of charge, remaining time to live, etc., along with the on-board component's

power requirement. Secondly, it must be mobile in order to explore its environment to find the possible energy resources, e.g., recharge station. And finally, to regain its health (energy) it should possess on-board recharging ability to recharge itself autonomously.

In reconfigurable modular robotics, the fact that there is no theoretical upper bound that limits the number of modules in an organism, increases the complexity of the evolved system. To keep or to sustain the autonomy of aggregated modules in the organism, the modules are required to establish a homeostatic control that maintains the equilibrium among the different system states. From the energy autonomy perspective, to deal with the non-uniform energy distribution, the autonomous modules are required to maintain a homeostatic control in an organism.

2.3.1.1 Energy Homeostasis

Homeostasis is a self-regulating process found in biological systems to maintain stability "while adjusting to conditions that are optimal for survival" (Britannica, 2009). In living cells, it is the innate ability of biological cells to maintain a consistent environment that is favourable for their survivability and well being. To sustain such a consistent environment the cells develop a strong interaction that enables them to deal with many kinds of anomalies that disturbs their equilibrium, with both the internal (intracellular) and the external (extracellular) environment (Dictionary, 2008). One example of homeostasis in nature is the regulation of body temperature in warm blooded creatures like human beings.

Modular reconfigurable robots that try to mimic the behaviour observed in nature, e.g., social insects, bees, birds, etc. require homeostasis among different system states when are docked or fused in an organism. For instance, the morphology that defines the structure of an organism, effects the utilisation of each individual module's capabilities docked at a particular position in the organism. Another promising application of homeostatic control in multi-robotic systems can be seen due to the non-uniform energy distribution among the modules in the organism. *Energy homeostasis* in artificial multi-robotic systems is a process that regulates the energy or power flow among the modules of an organism to achieve self-sustainability and self-sufficiency for a longer period of time without human intervention (Humza *et al.*, 2009). To regulate the energy distribution among the modules it becomes mandatory for every module to be aware of not only its own health status, i.e., the available and the required amount of energy, but also the status of the others in the swarm. Therefore, for online health status monitoring, every module requires a dedicated real time system to continuously measure and interact with the internal (intra-organism) and external (in a swarm) system perturbations that in turn helps to adapt its behaviour accordingly.

2.3.2 Energy Harvesting

As could be seen in Sect. 2.3.1, robots that move and operate autonomously must have access to some external source of energy to replenish their on-board resources.

Although in recent years considerable progress has been made regarding the output power capability and energy weight ratio of batteries (Linden & Reddy, 2002), the stored energy is still very limited compared to the power and energy demand of a meso-scale robot module. As is easily imaginable, this shortage becomes worse when energy-hungry, highly sophisticated sensor and processing systems are involved such as vision systems. In particular for self-reconfigurable robots, it is expected that in general a higher demand of sensing and processing power is required compared to "conventional" swarm robots that only move in 2D space, which will simultaneously raise the energy consumption. In order to increase autonomy it is hence wise to enable the robots to collect additional energy from the environment, which is often referred to as *energy harvesting*.

Depending on the particular robot modules and their environment of operation, different forms of energy harvesting for robots have been employed or discussed, like solar power (Landis & Jenkins, n.d.), (Boletis *et al.*, 2006), wind power, vibrational power, etc. (Wade & Gifford, 2007). Even energy harvesting methods based on fuelling organic substances like sugar (Wilkinson, 2000), sludge, and even flies (Ieropoulos *et al.*, 2005a) and slugs (Kelly *et al.*, 2000) by applying microbial fuel cells have been reported. The size of the latter harvesting devices were however considerable compared to the generated energy and it is unlikely that with current technology such systems will be successfully included into reconfigurable meso-scale robots: In (Ieropoulos *et al.*, 2005b) the MFC that performed best had an average output power of 45.5 μW over a period of 10 days and measured $6\,cm \times 7\,cm \times 5\,cm$. Nonetheless, used in a central charging unit (s. Sect. 2.3.2.1) that includes a fermenter as suggested in (Kelly *et al.*, 2000) this may become a viable solution.

When considering the energy supply of robots, size is an extremely important factor since technologies for energy storage and harvesting do not scale down that easily. For example, if a cube shaped robot module were shrunk in length by a factor x, the surface area would decrease by x^2, whereas its volume would be reduced by x^3. In a rough model, the energy consumption due to the weight of the module is more or less proportional to its volume. On the other hand, solar cells generate electrical energy approximately proportionally to their surface area. This implies that it will be much easier to power a small robot with solar cells than a larger one. Reports on energy scavengers confirm the scaling issue. Even though energy harvesting is being employed in more and more marketed mobile electronic devices (vibrational energy scavenging in wrist watches, thermal energy transformers for wearable sensors, etc.) many of these are not very useful for robotic applications in the meso scale. This conclusion is drawn from the fact that the reported appliances rely on extremely low power demand electronics and lack any actuator, a key component in robotics that due to physical limitations cannot be made energy preserving to an arbitrary level. It is certainly not possible to give exact numbers of robots' power consumption in general. There are too many factors that come into play, such as size and hence weight, processing power and grade of sophistication, principle of locomotion, number of sensors, and many more. In (Mei *et al.*, 2005) the author has tried to shed some light onto mobile, i.e. wheeled, robots of meso-scale. When looking at the given numbers (10–20 W) it is obvious that certain energy harvesting

methods at the current state of art are not able to provide the necessary power. When considering that self-reconfigurable robots will in most cases be more complex than ordinary swarm robots (sensor fusion for docking approach, docking mechanism, etc.) energy scavenging by integrating harvesting modules within the robot modules is even less promising because of the tighter size restrictions and the expected higher energy demand due to 3D actuation, when lifting other modules.

Nonetheless, photovoltaic cells are commonly being used for robot energy scavenging, in particular in space missions. The efficiency of solar cells not only depends on the specific technology being used (Si, Ga, thin film, crystalline, etc.) but also on the spectrum of light. Hence the cells are compared at a well defined spectrum, which has been standardised, e.g. IEC60904-3, at 1000 W/m^2. This irradiance prevails approximately at a cloudless noon during spring in central Europe. The highest reported efficiencies of PV modules – these are composed of several cells connected together – lie between 8.2 % (thin-film polycristalline Si) and 22.9 % (crystalline Si) (Green *et al.*, 2009). Consequently, a solar panel made of crystalline Si with a realistic efficiency of 15 % under ideal conditions will generate approx. 150 W per square metre on an average bright sunny day. In order to power a reconfigurable robot this may still not be enough, when taking a much smaller available surface area into account. But for a base or charging station to which the robot modules return to recharge this may be a realistic scenario.

2.3.2.1 Charge Stations

Placing charge stations into the work area to let the robots autonomously refuel themselves is one method that has been described often in literature to provide self-sustainability for robots. One example of a commercial robot that recharges by itself without human aid or prior request is the "Roomba" vacuum cleaner from iRobot Corp., MA (USA) (iRobot Corporation, 2007). The difficulty that remains is to find and identify these stations and to successfully dock to them. However, in self-reconfigurable robots this is a principle task that the robots must be able to solve.

The number of charging stations will be limited and presumably much smaller than the number of robot modules. In order to decrease the competition among swarm members, it is advantageous to design the robot modules in such a way that a robot that is docked to a charging station can offer additional charging ports to others via an "energy bus" and docking interface.

For robot modules that are supposed to operate in buildings, the straight-forward method is to actually plug into the nearest wall socket and recharge from the power line. In an office environment, wall sockets may be regarded as a ubiquitous form of "charging station". In this case, the robots would need to be able to detect and recognise wall sockets, which ideally would not need to be specially marked. So the robots must possess a tool to connect to them and convert the voltage levels appropriately, all adding to the complexity of recharging on the other hand. The feature of re-configuration may be particularly advantageous in harvesting energy from wall sockets, since in most cases a single robot module will not be able to harvest energy

from a socket because it simply cannot reach it. Hence it is forced to merge with others to a larger organism and scavenge energy by collective collaboration.

Another interesting aspect is to design charging stations in a way that these themselves are energetically independent, i.e. they incorporate means of energy harvesting, e.g. with solar panels. Since normally they are not required to move and can be stationary, it may be easier to design harvesting facilities into these rather than into the robot modules. Furthermore, one may think of robots foraging fuel that they themselves cannot burn, but only the charging unit can.

2.3.3 *Energy Trophallaxis*

In biology, the term *Trophallaxis* denotes the mouth-to-mouth (*stomodeal* (Korst & Velthuis, 1982)) and anus-to-mouth (*proctodeal* (Cabrera & Rust, 1999)) exchange and/or transfer of food among conspecifics, which is frequently observed at social insects, in particular bees and ants, and some vertebral animals. This sharing of food, however, is not limited to feeding as can be observed when parent birds feed their breed. In fact, it seems that in many cases food is not passed on for the pure purpose of food (energy) transfer but for communication reasons (Korst & Velthuis, 1982), (Camazine, 1993). Consequently, in robotics trophallaxis has been applied for both, energy homeostasis in a swarm (Melhuish & Kubo, 2007), and as a means of non-centralized communications within a large robot swarm colony (Schmickl & Crailsheim, 2008).

The latter makes use of the "food" gradient that appears provided dissipation is present, which indicates to others the direction and yield of a remote energy source. Trophallaxis is in particular useful for insect colonies that need to regulate the internal state of the colony, e.g. the protein supply in the bee hive, and do not have any centralized processor such as a brain and lack a communication path way like the nervous system. Hence, for swarms of disjoint robots it can similarly aid in regulation and self-organisation. A reconfigurable robot in the context of this book, however, does have a dedicated communication channel and may in some settings

(a) (b)

Fig. 2.21 Trophallaxis in social insects: **(a)** Trophallaxis between two honey bees – mouth-to-mouth food transfer (with the kind permission of Eric Tourneret). **(b)** Trophallaxis between ants – mouth-to-mouth food transfer (with the kind permission of Alex Wild).

also provide a central "brain" by using distributed processing in only a few of specialised modules. It is even questionable if the term trophallaxis is valid at all *within* a robotic system consisting of merged[7] robotic modules that simply share their energy on a common energy bus. Trophallaxis may consequently not have the same significance in reconfigurable robots, unless these themselves are a member of a swarm of *organisms*. Generally, in a swarm of self-reconfigurable robots one will find single, disjoint robot modules and robot organisms of different morphologies consisting of a variable number of merged modules. If two such robot units, may they be single or merged, dock together for the only sake of energy transfer, we regard the term trophallaxis as being a valid analogy.

Most robots today are electrically powered and store their energy in batteries or capacitors. But, there are examples where the energy storage is done differently, e.g. by using fuel cells and storing the energy as methanol or even as organic fuel (fly eating robots). The technology how the exchange of energy is achieved will consequently vary with the employed storage method. Nevertheless, in the meso-scale, probably the most simple way of sharing energy among docked robot cells may be through a common electric bus to which each of the modules have access.

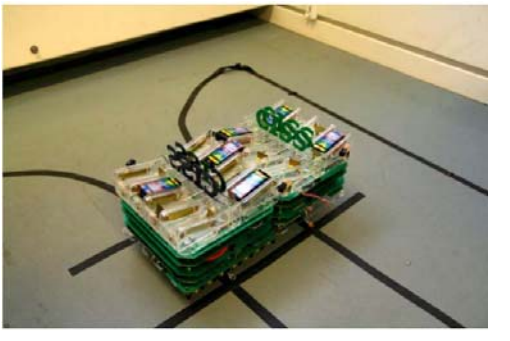


Fig. 2.22 Artificial energy trophallaxis: Two CISSbots in a state of exchanging battery (with the kind permission of Trung Dung Ngo).

Trophallaxis based on recharging the other's battery is quite time consuming (considering today's battery technology up to several hours) and inefficient from an energetic perspective. Hence, Ngo et. al have devised mobile swarm robots that exchange energy through swapping battery packs that they carry on their top out of a specifically designed battery holster unit that can hold up to 8 ejectable batteries (Ngo & Schiøler, 2006). The idea behind it is to increase sustainability of the swarm since individual robots do not need to permanently interrupt their current tasks and return back to the refuelling station.

It is important to stress that by trophallaxis a swarm of robots does not gain energy. In fact, overall it loses energy since the efficiency of energy transfer will always be less than perfect. However, a swarm may increase its autonomy through trophallaxis since the swarm can take over the role of a "common stomach" levitating any peaks and dips of energy supply and assuring a more evenly distribution of supply.

[7] Although the robot hardware cannot literally merge, this term is being used nevertheless to underline that the modules do not only physically aggregate and bond, but also unify logically – they become one larger entity.

2.3.4 *Energy Sharing within a Robot Organism*

We have seen that in a swarm of disaggregated robotic modules the sharing of energy is a common means to ensure energy homeostasis over the swarm – be it that multiple robots share energy resources e.g. recharge stations, located in the working space or by direct exchange or transfer of energy from one robot to the other, i.e., trophallaxis. In an artificial organism composed of individual robotic modules, energy sharing becomes even more vital due to the uneven workload at each individual that varies considerably. For example, in an artificial lizard shaped organism as shown in Fig. 2.26(d), modules that are responsible for the hip joints will most likely require more power than the modules present or docked in the head. Furthermore, due to the non-uniform energy distribution and variable workload the longevity of such a creature would hence be limited by the weakest of its docked module if energy would not be shared among the robotic modules that constitute it. Moreover, there may be incidences where even a fully charged robot cell cannot perform a certain task without the energy support from fellow robots simply because of the physical limitations of its battery pack. Most of the reconfigurable robots that have been reported do consequently not only have a docking unit to let them mechanically hook themselves together. In addition, they have an interface for the energy exchange among the modules. Different state of the art reconfigurable modular robots such as ATRON (Jorgensen *et al.*, 2004), SUPERBOT (Salemi *et al.*, 2006), incorporate energy sharing features in their system design. But none of them highlights or investigates the energy management issues that may arise in a multi-robotic organism. Fig. 2.23 shows the power sharing circuit schematic of the SUPERBOT module. The system design includes six docking faces, and on each docking side there is a switch and a diode combination that controls the direction of flow of current on the power lines among the modules (Salemi *et al.*, 2006). By default, the switch that connects the battery with the charger leads remains ON while the rest of the switches that are present on each of the six docking faces stays open. In this configuration, the diodes on each face provide an alternate path to the current flow on the power lines, thus allowing the current to flow into the system when a module is docked either to a recharge station or to a module. The docking switches on the six docking faces that are controlled by an onboard controller enable a module to share its battery power in the respective

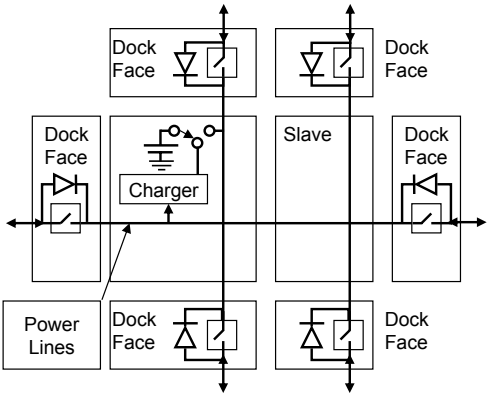


Fig. 2.23 Power sharing circuit schematic of SuperBot (with the kind permission of Wei-Men Shen).

strand. Since the SUPERBOT power sharing design does not include any fault tolerance feature, therefore even a slight deviation from the normal on the power lines could be disastrous for the whole organism. For example, a short circuit in a module of an organism could shut down the organism's power lines/bus.

2.3.5 Energy Management

Energy management refers to planning and controlling the generation, storage, distribution and dissipation of energy in a system. In order to increase the autonomy of a robotic device, energy management is probably the most important factor. From the energy management perspective, multi-robotic systems can be broadly classified into systems with "*harvesting*" and "*non-harvesting*" capability. The former system refers to those that own a dedicated energy harvester, e.g. solar panels, etc., while a non-harvesting system lacks an on-board energy harvesting mechanism and therefore is completely reliant on external sources for energy replenishment, e.g. charge stations.

In biology, another important factor known as, the dynamic energy budget (DEB) model describes the process by which an individual organism acquires energy through digestion in order to utilise it for their growth, reproduction, and survival (Nisbet *et al.*, 2000). It also quantifies the energetics of an individual as it changes during life history. This concept can be taken over to multi-robotic organisms where the energy management in a swarm of reconfigurable modular robots becomes essential when considering the autonomy of every individual. The DEB theory not only emphasizes the energy gathering or the energy acquisition but also its efficient utilization by means of task planning and real time task scheduling.

In the context of this chapter, one important aspect of the energy management of a multi-robotic organism is to sustain the aggregation and collaboration of modules in an organism that is only possible when the system design incorporates fault tolerance features. The fault tolerance is a design feature that enables a system to operate properly or within the defined limits in the presence of failures or perturbations. From the energy homeostasis perspective, as described in sec. 2.3.1.1, the fault tolerance feature of the system design must enable the modules to keep the organism's power bus alive at all times. In order to further improve the modules' fault tolerance ability an immune system concept found in biological systems can be incorporated into the system design. To keep the living beings healthy, the inherent immune system provides the body a defensive mechanism against different diseases that identifies and kills the pathogens. In the similar fashion, an artificial immune system (AIS) that mimics the biological immune system behaviour can be added to a robotic module's system design. The on-line on-board artificial immune system in conjunction with the dedicated real time energy management ensures the operation of the organism's energy management within the steady state. For further details on how AIS is incorporated into the system design, please refer to Sect. 4.4.

2.3.5.1 Methods of Energy Management

The energy management system (EMS) can be realized as a control system with an active feedback mechanism that monitors the environment (energy resources) and the system load (power consumption) to adapt its behaviour accordingly. The feedback mechanism enables the system to maintain its operations within a steady state while operating in a dynamic environment.

In purely battery powered robotic systems, where the only energy source is the on-board battery pack, the essential task of the energy management is to reduce the overall system power consumption without compromising the system's performance to maximise its operation time. Several algorithms have been proposed in the literature that try to reduce or minimise the power consumption without affecting the overall system performance at different levels, for example, Adaptive Voltage Scaling (AVS) (Elgebaly & Sachdev, 2004), Dynamic Power Switching (DPS) (Benini & Micheli, 1998), Dynamic Voltage/Frequency Scaling (DVFS), adaptive shut down mechanisms, and others. Another important method that helps to reduce the system power consumption is by "task prioritization" and "real time task scheduling". For instance, upon the indication of low battery voltage the "energy refuelling task" gets the highest priority or the scheduling of tasks based on their power requirement. An example of such a system is the "Roomba vacuum robot" (Tribelhorn & Dodds, 2007). It is a purely battery driven robot that autonomously moves to the nearest recharge station on the indication of low battery voltage.

The energy management system in an artificial robotic system incorporating an energy harvester is further responsible to gain long term self-sustainability by adapting the system behaviour with the availability of energy resources in the environment. To accomplish this, the energy management system can also prioritize the utilization of harvested energy. Several EM algorithms have been proposed that prioritize the utilization of harvested energy in different scenarios. For example, if the on-board battery pack is nearly drained, then a major portion of harvested energy will always be used for battery recharging until a safe state is reached. The stored

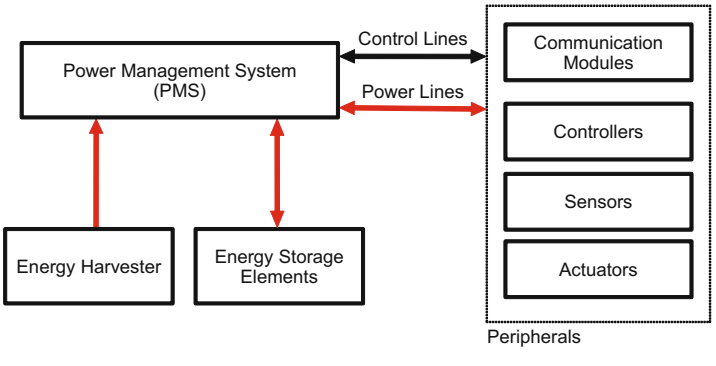


Fig. 2.24 Block diagram of a robotic module with on-board energy harvester.

energy is then being used when the harvested energy becomes insufficient to drive the peripherals. Examples of such systems are the "EcoBot-I" (Bennetto, 1987) and "SOLERO", a solar-powered exploration rover (Michaud *et al.*, 2002). Figure 2.24 shows the block diagram of a robotic module that owns an on-board energy harvesting unit. The system components include an *energy harvester*, *energy storage elements*, an *energy/power management system* (PMS) and essential *system peripherals*.

A fundamental difference between the harvesting and non-harvesting system designs is that in the prior, the recharging of on-board battery packs can be carried out meanwhile the energy is being harvested whereas in the latter system, it requires a considerable amount of time and effort to replenish its energy.

Dynamic Power Management

Dynamic power management (DPM) is a design methodology that dynamically controls the system's electronic components in order to minimise the overall power consumption without affecting the system functionality and performance (Lorch & Smith, 1998; Benini & Micheli, 1998). The fundamental assumption behind the applicability of DPM is the non-uniform or variable workload during a system's life time that can be predicted in advance with a certain degree of confidence. The application of dynamic power management is not just limited to the minimisation of power consumption. Rather, in the context of collaborative reconfigurable multi-robotic systems, it deals with the efficient utilization of energy in multiple modules that are physically docked together. Following are the proposed dynamic power management techniques:

- ***Proactive Power Management*:** also known as predictive power management. In real time systems it becomes mandatory for a system to predict or estimate its future power requirements to ensure the autonomy and the self-sustenance of an artificial system. The predictive power management utilizes its past experience, i.e. the knowledge of its prior energy consumption, to predict the future consumption under different load conditions. Let the variable A represent the overall instantaneous robot module's power consumption, while B represents the system load (components that are in ON state or running). To predict the future power demand, $p(A)$ represents the probability that the system consumes A amount of power at a particular instant, and $p(B)$ represents the probability of having system load B. One way to estimate the module's future power requirement from the current and past observations is by employing Bayes' rule (Howson & Urbach, 1993). It uses the likelihood function and the prior probability to maximise the posterior probability distribution, i.e.,

$$posterior\ probability = likelihood \cdot prior\ probability$$

  In our case, the conditional probability $p(B|A)$ represents the likelihood of system load B with the given power consumption A. The conditional probability

$p(A|B)$ represents the posterior probability that the system power consumption is A when it has B load. Using Bayes' rule,

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}, \tag{2.2}$$

where $p(B)$ is the prior or marginal probability of B, and acts as a normalizing constant.

These early power consumption estimates in the presence of dynamically changing load conditions enable an autonomous system to steer its behaviour according to environmental changes.

- ***Reactive Power Management*:** is the innate ability of an artificial system to react to instantaneous changes occurring in the system. An important difference that distinguishes it from the proactive power management is its instantaneous response that does not require any learning cycle. A hardware current limiter is one such example that limits the flow of current through a system's power lines. In case of an error, e.g. short circuit, the current limiter automatically limits the current flow to avoid any system breakdown or major damage. Another important application of reactive power management can be seen in case of unused system components. For instance, when there is no application with an open socket, having an Ethernet controller powered up is a waste of energy. Therefore, with respect to energy consumption, it will be wise to turn it 'ON' only when the system is ready to receive or transmit data.

In order for a power management system to be effective, it should be given the ability to turn on or off diverse peripherals. The activation of these peripherals can be used to define a number of power saving modes. In SYMBRION/REPLICATOR, the following major modes have been defined:

1. ***Active mode*:** All major components can be turned ON, such as actuation, processing, sensing, and communication.
2. ***Doze mode*:** Typically the actuation and different sensing components are often power hungry components, therefore in doze mode they will be completely shut down. The processing (microcontrollers and microprocessors) and communication modules are turned down to a power saving duty cycle mode.
3. ***Hibernation mode*:** All components, including processing and communication modules are switched off. Re-activation of the module is only possible by externally applying sufficient energy onto the energy bus.

These different power saving modes can be used by the energy management system to prolong the operation life of the robotic system. Below, an example is given when these modes are entered in a pre-engineered approach.

1. ***Self-sufficient state*:** In this state, a robotic module is regarded as being "healthy" and the active mode is chosen. The individual robot has enough energy to perform different tasks in the swarm. With its available energy it is also able to return to the recharge station in the environment for refuelling. The operations in this state last as long as the module's on-board energy is greater than the energy required

to move and successfully dock to a recharge station. Let E_a represent the amount of energy available on-board, and E_{rq} be the amount of energy required to locate and dock to a recharge station in the vicinity. The condition for the self-sufficient state is thus:

$$E_a > E_{rq}, \tag{2.3}$$

where the remaining energy $E_{rem} = (E_a - E_{rq})$ is then used by the individual to perform its dedicated tasks or to help fellow modules in the swarm. The variable E_{rq} is defined as system and environment dependent. To sustain their autonomy, every individual in the swarm continuously learns its appropriate values by monitoring their load conditions with the changing environment. Also, a safe margin for unexpected occurrences should be considered.

2. ***Self-sustenance state*:** indicates the state of a module when

$$E_a \leq E_{rq}, \tag{2.4}$$

the available on-board energy falls below the safe margin of the self-sufficient state. The module's surviving gets the highest priority, i.e., recharge itself before it runs out of energy. Hence, it will either dock to a recharge station autonomously or seeks help from a fellow robot to reach a recharge station.

3. ***Doze state*:** When the available energy drops further, doze mode will be selected to save its remaining energy. A module is forcefully entered in doze mode when the available energy is not expected to be sufficient any longer to reach a charging station or a fellow robot that could help. That is,

$$E_a < a \cdot E_{rq}, \tag{2.5}$$

where the scaling factor a is again system and environment dependent.
In doze mode, a module behaves passive and from time to time tries to identify other robots in its vicinity through its sensors in order to call for help.

4. ***Hibernation mode*:** is analogous to the doze mode, in which a module by choice after shutting down all its components enters into the sleep state. A module may choose to enter in hibernation mode either it completely ran out of energy or energy preservation is desired. The significant difference in the two energy saving modes is, in doze mode a module can still communicate with other modules in the colony that shows its existence where as in hibernation mode it may be treated as an obstacle or a dead robot that can only be brought back to life through external donation of energy.
5. ***Dead Mode*:** indicates the fatality of a module. The dead mode in the life of an individual is attained only when it loses its mechanical stability or when it loses its core processing units due to any electronic system failure.

Fig. 2.25 shows the state machine that links the different states during the life cycle of a SYMBRION/REPLICATOR robotic module, as described above.

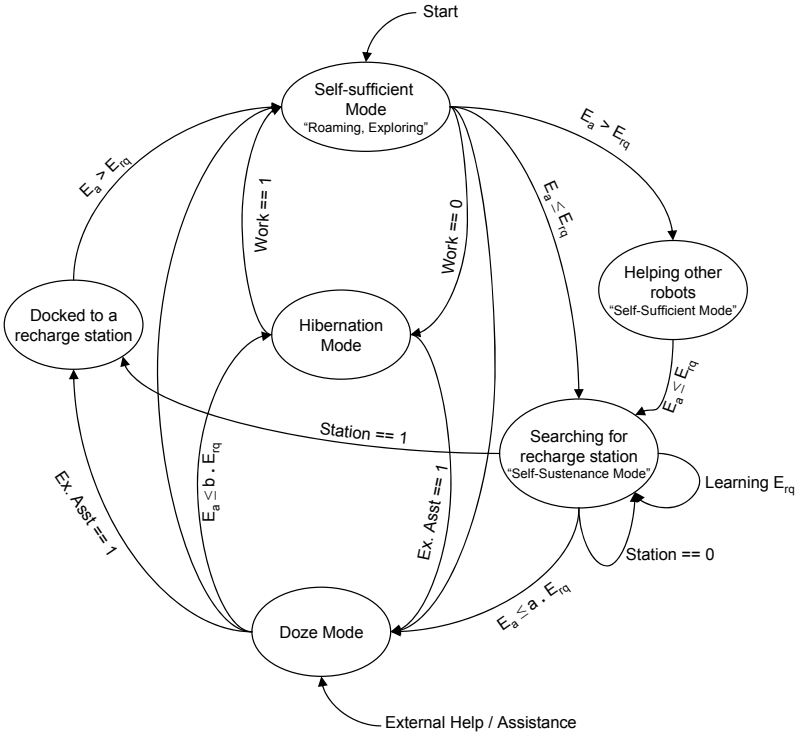


Fig. 2.25 State machine that shows the different states during the life cycle of an autonomous self-reconfigurable SYMBRION/REPLICATOR module.

Energy Management of Multi-Robotic Organisms

Like living organisms the most vital component for an artificial robotic module is the energy supply. From the self-sustenance perspective, the energy/power management of an autonomous robotic module has to only deal with its own survivability. On the other hand, in an organism with multiple modules having different energy levels and varying power requirements the ultimate objective of the energy management is to somehow sustain the dynamic collaboration of modules in the organism. In order to do so, the collective energy management that emerges from the individuals' energy management must fulfil the following criteria:

- To deal with non-uniform energy distribution in the organism it must be capable of maintaining the energy homoeostasis in an organism, i.e., balancing the energy reserves among the modules, either by load sharing, recharging the weaker modules from the healthy modules, or by changing the morphology of the organism,
- It must be able to regulate the energy flow among the modules to avoid overload conditions,
- provides fault tolerance features to withstand system perturbations,

- also, it must be capable of monitoring the organism's health status from the individual modules to steer and control the intra-robot actions in the organism.

In a similar way as the mechanical construction of the docking unit (male/female vs. genderless, robustness) constrains the morphologies of a reconfigurable organism, so does the implementation of the docking contacts and the finite current carrying capability of the "energy bus" limit the overall performance of the organism. Every contact may conduct electrical currents up to a certain amount until the voltage drop due to its resistance gets too high. If too much current is passed through the contacts they may even get destroyed. That means there exists an upper bound to what size an organism can grow that is dictated by the limitations of the energy sharing bus. It is not easy to determine this upper bound since it is influenced by a large number of factors such as maximum power requirements of a single module, the number of cells in a chain at a particular energy port, the load distribution within the organism, etc. The energy management must take these factors into account.

Example 2.1. Docking contacts:
Let us consider a group of heterogeneous and homogeneous robots that are docked to each other in the form of a chain-like structure shown in Fig. 2.26(a). To reduce the power losses across the docking contacts the proposed power management scheme for the robots uses 6 serially connected lithium polymer (LiPo) cells that deliver a nominal voltage of 22.2 V with approximately 900 mAh of charge capacity. In the current design, the proposed docking contacts can withstand a maximum current of up to 8 Amp. If one battery pack would need to be recharged with a current of 500 $mAmp$, then theoretically, no more than 16 modules can be recharged simultaneously in an organism, with the said current carrying capacity of the power bus.

In order to increase the robustness of the reconfigurable system it must be possible to turn off the individual strands of the energy bus. Otherwise, a fault such as a short circuit condition in one of the cells could easily destroy many more of its fellow cells. Unfortunately, every switch and current sensing component placed into the bus adds additional losses and voltage drops. One way to sooth the problem is by maximising the voltage level on the bus. The electrical power conducted by the bus is the product of the current through the bus times the voltage across the bus. In other words, when increasing the bus voltage the current may be decreased to maintain the provided electrical power. Since the resistances of the contacts and the other components in the bus are constant, the various losses on the bus decrease correspondingly.

Example 2.2. Energy sharing:
As stated earlier, the energy sharing among the self-reconfigurable modules is achieved by establishing a wired interconnection among the modules that are docked together in an organism. Fig. 2.26 shows the different configurations of SYMBRION/REPLICATOR modules forming biologically inspired organisms.

For collective actuation the modules in the organism are required to share their resources, which in turn demands a dedicated "dynamic resource management mechanism" to balance and control the organism's resources not only with the joining

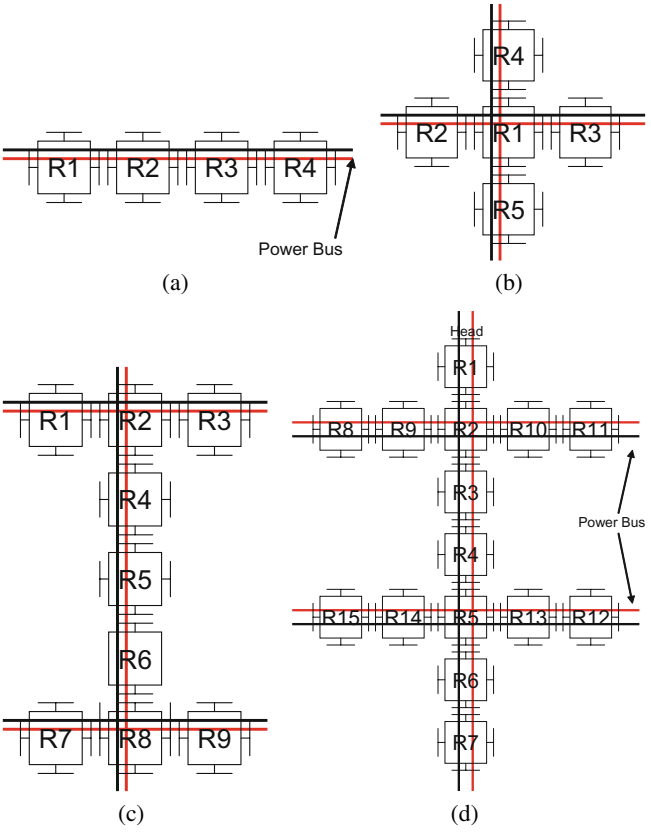


Fig. 2.26 Block diagram of self-reconfigurable robotic modules docked in different configurations.

and leaving of modules but also with the changing environment. From the power management perspective, one way to balance the uneven energy distribution among the modules of the organism is by establishing energy homoeostasis described in Sec. 2.3.1.1. For that, each individual module must be aware of energy distribution among the module in the swarm. To accomplish this, a concept of residual energy scan or *eScan* already known in wireless sensor networks (Zhao *et al.*, 2002) can be applied to a swarm of reconfigurable robot modules. The residual energy scan is a process by which the network nodes collect residual energy information available at each node distributed in the network. The information is then utilized for different purposes, e.g., for dynamic load sharing, to predict sudden system failures in the network, etc. With the periodic information exchange in case of failure detection the faulty or weak nodes can be removed from the network.

The dynamic power management of the organism that helps in establishing the energy homoeostasis among the modules can be realized in two forms namely the "centralized" and the "decentralized" topology.

- ***Centralized power management:***
  The centralized power management mechanism works by selecting a group of healthy modules based on their state of charge in the organism which can be called "master modules". This multi-master architecture of modules in the organism that are responsible for dynamic resource sharing, i.e., energy homeostasis, adds redundancy which in turn increases the organism's fault tolerance ability. These master modules in the organism maintain a data base containing the residual energy status of each module along with their instantaneous power requirement. The residual energy information is then used not only to maintain energy homeostasis but also to adapt different morphologies.
- ***Decentralized power management:***
  In a multi-robotic organism, with centralized power management topology, due to the variable or non-uniform workload it becomes difficult to elect and re-elect the master modules in the organism. To overcome the said limitation, the distributed nature of decentralized power management allows every module to be aware of energy distribution in the swarm. For this every module in the swarm keeps a database containing the residual energy information of each module. The decentralized power management scheme on one hand increases the computation at each module, on the other hand, empowers every module to decide at its own.

Energy Management System

To demonstrate the energy management principles in self-reconfigurable modular multi-robotic systems we have chosen the SYMBRION/REPLICATOR modules as an example. Fig. 2.27 shows the block diagram of the power management system for an individual module. The power management design incorporates 6 lithium polymer cells that provide a nominal voltage of 22.2 V with approximately 900 mAh of charge capacity, a battery management module, a step-up converter for recharging, core processing components, system peripherals and several software controllable switches with current limiters. To enhance the system fault tolerance the onboard electronics is segregated into two blocks, namely the, "core processing components" and the "high voltage (22V2) peripherals". The core processing components that include micro-processors, micro-controllers, etc, are continuously powered from the battery source while the high voltage peripherals can be either powered from the onboard battery pack or from the power bus. The "battery management module" (BMM) is composed of an analogue front end (AFE)[8] and a low powered MSP microcontroller[9]. It is used to extract and control the critical battery pack parameters e.g., state of charge, charging/discharging, cell balancing, etc. A step up converter in the system design adds the energy trophallaxis feature – recharging one module from a fellow robot's battery pack. The "power manager" controls the system behaviour by processing the low level information to monitor and control the switches

[8] BQ77pl900, 5-10 cell Li-ion Battery protection and AFE by Texas Instruments (TI).
[9] MSP430F2410, 16-bit Ultra-Low-Power Microcontroller by TI.

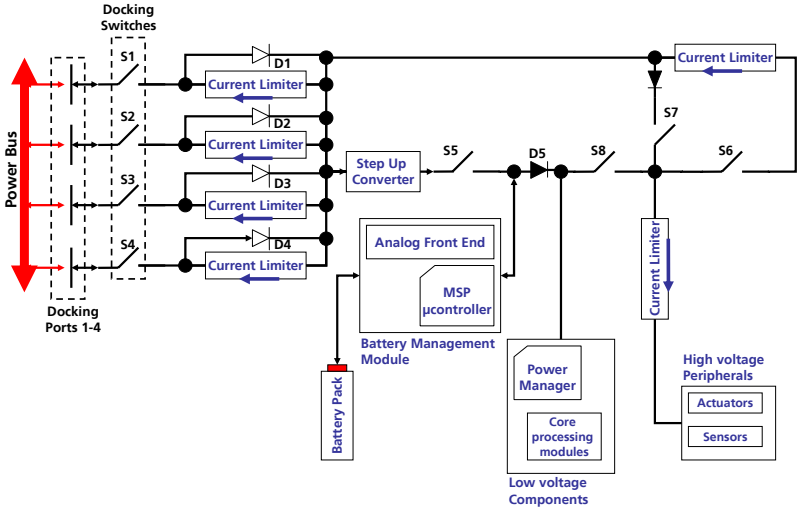


Fig. 2.27 Block diagram of the proposed power management system for an individual robot.

and other system states. The current limiters in series with the docking switches $S1 - S4$ are added to further increase the fault tolerance feature of the power management system.

To understand the purpose of the docking switches consider a group of modules docked to each other forming a star topology as shown in Fig 2.26(b) with all the docking switches in closed state. Suppose a fault in the form of a short circuit appears at robot $R3$. Due to this short circuiting at $R3$ a high inrush current from the power bus flows into $R3$. This inrush current that exceeds the rated current limit momentarily shuts down the current limiter present at $R1$ facing $R3$. The malfunctioning of the system component is then detected by the respective power manager in order to adapt its response accordingly. In the organism such a fault tolerance feature enables each module to trace and get rid of faulty modules in the organism.

The switches $S5, S8, S7$ and $S6$ in the system design are included to control different system states. The switch $S5$ acts as a "charging switch" that turns ON and OFF the recharging of the on-board battery pack from the power bus. $S8$ connects and disconnects the secondary peripherals to/from the on-board energy supply, whereas the switch $S7$ in ON state powers the high voltage peripherals from the power bus. At the end, switch $S6$ acts an "energy sharing" switch that allows the flux of the system battery power into the power bus. Table 2.6 describes the different operating modes of the power management system.

2.3.5.2 Energy Aware Self-Organization of Self-Reconfigurable Modular Robots

The concept of "ecological balance" introduced by Pfeifer (Pfeifer, 1996) in the context of a set of "design principles for autonomous agents", describes the relation

Table 2.6 Truth table describing the functions of switches $S5, S8$, and $S7$ in the proposed power management system.

S5	S8	S7	Description
0	0	0	Only the core processing components are running from the on-board battery power
0	0	1	Now, the high voltage peripherals (22V2) drive themselves from the power bus
0	1	0	The core processing components and the 22V2 peripherals both consume power from the on-board battery pack
0	1	1	The core processing components and the 22V2 peripherals both consume power either from the on-board battery pack or from the power bus
1	0	0	In this configuration the step up converter provides charging voltage, i.e., 25V2, to the battery pack and to the core processing components
1	0	1	In this configuration the step up converter provides charging voltage to the on-board battery pack and to the core processing components whereas switch S7 gives the power to 22V2 peripherals from the power bus.
1	1	0	Fault condition: Turning S5 and S7 ON *overloads* the step converter. Since, now both the core processing modules and the 22V2 peripherals are hooked to the output of the step up converter.
1	1	1	Fault condition: Turning S5 and S7 ON *overloads* the step converter.

between morphology, material and control of an autonomous system. In collaborative multi-robotic systems, the adaptive behaviour of autonomous modular robots in a swarm is not just limited to the adaptive control evolved over time but also greatly depends on the morphology of these modules when are docked in an organism. Self-organization that is the inherent ability of a self-reconfigurable multi-robotic system, refers to *both* the dynamic adaptation of morphology and the control architecture with the changing environment (Lund *et al.*, 1997). The morphology that defines the structure of an organism i.e., the arrangement of modules, plays a significant role in the behaviour of an organism and also in defining the capabilities that result from the collective actuation of aggregated modules. It is the morphology that defines the degree of freedom a module can exercise in the organism.

As mentioned earlier, an important component that effects the autonomy and hence the morphology of an organism is the energy distribution among the modules. *Energy aware self-organization* is a phenomenon that defines the aggregation of modules, i.e., their physical position, within an organism with respect to their residual energy level. This way the modules in the organism are arranged or rearranged that helps them to establish energy homeostasis by sharing the residual energy among the weaker and healthy modules.

From the system design perspective in a reconfigurable modular robotic organism where the upper limit of module count in an organism may not be defined, the increasing number of modules in the organism imposes different constraints that limit the efficiency of the power management system, as briefly highlighted in Sect. 2.3.4. In such a scenario, if the particular morphology of the organism is absolutely desired then the modules must reorganize themselves in the organism with

respect to their residual energy level to continue their operations in the organism. Again, for re-organization each module in the organism must be aware of not only its own but also the health status, i.e., residual energy and the power requirement, of other modules, that can be easily accomplished using an energy scanning process, as described above.

When re-organizing, a critical situation arises when a large sized organism has to change or adapt a new morphology. Empirically, the docking and undocking of modules in the organism consumes a considerable amount of power. Therefore, in such a scenario, before adapting the new morphology, an organism must also consider the overall *energy required to re-organize* (undock and dock) its modules. In a simplest scenario, consider an organism with n modules docked to each other forming a chain shaped organism as shown in Fig. 2.26(a). Consider E_{undock} represents the amount of energy required by a module to undock to a module and E_{dock} represents the amount of energy required to dock to a module. For the sake of simplicity we are not considering the amount of energy required for realigning the modules before docking. An organism with n modules requires $E_{undock} \cdot (n-1)$ amount of energy to undock all its modules and for re-docking again requires $E_{dock} \cdot (n-1)$ amount of energy. Altogether, the total energy E_{Total} required to an organism to change or adapt a new morphology is thus obtained as,

$$E_{Total} = (n-1) \cdot (E_{undock} + E_{dock}). \tag{2.6}$$

This becomes an important factor when a large sized organism requires to re-organize its modules. An optimal solution to the problem can be the decomposition of a large organism into multiple smaller organisms. This way the energy and resource management in smaller organisms becomes much easier and less complex to handle.

2.4 Modular Robot Simulation

Lutz Winkler, Heinz Wörn

At the beginning of a project hardware is often not available or not sufficient sophisticated, therefore simulation tools are essential. Later in a project, when such hardware is available and experiments with real robots can be realised, the simulator will still be required as such experiments are very time consuming and expensive. It helps to evaluate different robot configurations and to identify important aspects that have to be taken care in robot design.

The particular tasks a simulator for modular robots has to cover are:

1. *Single Robot Control*. Each single robot needs to is an entity and has to be simulated separately to help developing swarm algorithms. The single robot can be simulated dynamically, but does not need to be simulated that way. A 2D-simulator is often sufficient enough.
2. *Organism Motion*. When the robots are connected together, they build an organism which also has to be simulated. As the organism often accomplishes

movements in the 3D-space, a 3D-simulator, that either simulates the movements kinematically or dynamically is essential.
3. *Evolving Organisms Structures*. Like natural evolution, different organism configurations will compete each other in every generation. The best configurations will be used for the next generation to form slightly different organisms. Over time the population improves, and eventually a satisfactory organism will be evolved that can be used on the real robot. The simulator has to provide functions to support organism evolution.
4. *Developing and Evaluating Control Algorithms*. Like organism structures, organism control algorithms can also be evolved and the simulator needs also provide functionality for evolutionary control algorithms.
5. *Creating Test Scenarios towards the Grand Challenges*. Grand Challenges are defined to identify the progress of the projects. To evaluate the quality of the software, the simulator has to support the Grand Challenge scenarios and therefore needs to run very stable while simulating hundreds of robots.

According to the different tasks such a simulator has to fulfill, a lot of requirements are arising. It has to be able to simulate large robot swarms as well as complex organisms containing lots of degrees of freedom. Additionally, sensors and actuators of each robot need to be simulated as accurate as possible. Detailed sensor and actuator models are therefore important. The algorithms and programs that are later meant to run on the robot platform also have to run in the simulator without major changes in the source code. Different environmental features, either artificial like electrical sockets or switches or natural like temperature or humidity need to be implemented in the simulator to support different scenarios. In large project where many partners are involved, a modular design of the simulator is mandatory, so that for each experiment a different configuration can be used. This modular design will also help to distribute the simulation over different computers in a network. Such a distributed simulation is also strongly required, as the requirements to the whole simulator are too high to be executed on a single computer.

2.4.1 Simulation Environments

We will first examine different simulation environments and APIs to figure out which platform is the most suitable for developing the simulator. The underlying software platform should cover all the aspects mentioned above and as the projects are open source, open science and open hardware, an open source solution for the simulator is also preferable. The first choice would be to evaluate general simulation environments that are often used in robotics or in exploring swarm behaviors. The most common open source simulation environments are:

NetLogo NetLogo is a multi-agent programming language and integrated modeling environment that is well suited to explore emergent phenomena and swarm behaviors. However, as the agents need to be written in the NetLogo language, it is not possible to use the same program on the robot

without having a NetLogo interpreter included. Additionally, the lack of a dynamics simulation makes it impossible to simulate the organism movements accurately.

Breve The breve Simulation Environment is designed to simulate multi-agent system and artificial life in three dimensional space including simulating dynamics. However, like NetLogo a script language is used to describe the controller programs, which slows down the simulation and does not fit our idea of reusing the code on the real robot. Additionally breve only supports the Open Dynamics Engine, which does not support hardware acceleration on the graphic processor. Distributed simulation is also not supported.

Player The Player project combines the Player network and the Stage and Gazebo robot platform simulators. It is deployed in many robotic research projects. While Stage is a 2D simulator, Gazebo is a 3D dynamic simulator. For the dynamics simulation the Open Dynamics Engine (ODE) has been chosen as well. Although Player supports the distribution of the simulation over a network through a hardware abstraction layer, it does not support a state of the art distributed simulation, like described in (Fullford, 1996). Such a distributed simulation is very helpful as different simulations can interact together, e.g. an environment simulator which simulates the distribution of a gas can interact with the modular robot simulator. This way different sensor simulations, which are computational intensive can be developed independently and can run on different platforms. Additionally, the ability to use and switch between different dynamic and non-dynamic simulators at runtime need also be implemented as the user currently need to decide before starting the simulation whether he wants to run a fast 2D simulation (Stage) or a 3D dynamic simulation (Gazebo). Such a feature is however important as not all robots need to be simulated dynamically (robots in swarm mode for example can be simulated with a 2D simulator most of the time).

All these simulations do not support dynamics simulations that support hardware acceleration nor support state-of-the-art distributed simulation. These are key features that we need for successfully simulating modular robots, their actuators, their sensors and their controllers. Another solution would be to use a simulation environment from other modular robot projects, such as:

Molecubes Molecubes is an open source project which has the aim to design a cheap and robust modular robot system for everyone. It provide design files, instructions and software to encourage people to build their own Molecube robots. Within this project a simulator based on the Ogre game engine and the Nvidia-PhysX dynamics engine has been developed. This simulator however is too much based on the Molecubes robots. As the Molecubes need to be coupled manually, the models in the simulator are assembled at the beginning of the simulation. Coupling the robots during runtime is not possible. The Molecubes also

do not have actuators that allow them to move separately from the organism, i.e. they do not have wheels. Therefore a swarm mode is also not implemented in the design of the simulator. These issues can be added in the simulator as it is open source, however as it is based on the Nvidia-PhysX engine the dynamics engine cannot be adopted to the requirements for the modular robot simulator as it is closed-source. Also, there is no interface for a distributed simulation provided, which we need strongly.

S-Bot — The 15 *cm* small differential wheeled S-Bot robot is designed to study inter-robot communication and team work. It is equipped with a lot of sensors, including proximity sensors, sound detector and an omni-directional camera. Within this project a simulator based on the Vortex physics engine has been developed, where all sensors have been implemented. Another nice feature is the dynamic model switching, where according to the computational power and the current requirement, different detailed models of the robot can be loaded dynamically. Though it is a good simulator it is too much designed for the s-bot and needs huge modifications for our needs. Additionally, the Vortex dynamics engine is commercial and stays in conflict with the open source thought we like to pursue. Like the Molecubes simulator it also does not support distributed simulation, which is essential for our project.

Webots — Webots is a commercial development environment to model, program and simulate mobile robots. It has been developed together with the e-puck robot and has a huge library of sensor and actuators. Programs that have been evaluated in the simulation can be transferred directly to the robots. However as it uses ODE, the computation for the dynamics simulation cannot be transferred to the graphics card and distributed simulation is also not supported.

All of the above mentioned simulators do not fulfill the requirements that are needed for a simulator of modular robots. Therefore, we decided to develop a new simulator that covers all the aspects needed for such a simulator. The first question that arises then developing a new simulator is the underlying framework. As we do not have the manpower to create a completely new simulation environment, we need to use a simulation framework that is already available. Common open-source 3D rendering engines that come in the closer choice are:

Irrlicht — Irrlicht is a high-performance realtime 3D rendering engine which focuses on features for visualization like dynamic shadows, particle systems, character animation and collision detection. However, neither a physics engine is supported which needs to be integrated, nor a distributed simulation.

Ogre — Ogre is another very sophisticated rendering engine with lots of features. It also supports different dynamics engines like Bullet, ODE or

Nvidia PhysX, however interfaces to distribute simulations are not provided and still need to be added.

Delta-3D Delta-3D, is an open source game and simulation engine that uses other well known libraries like OpenSceneGraph (OSG) for visualization, Open Dynamics Engine (ODE) and Nvidia PhysX for dynamic simulation. They also offer a physics abstraction layer. That way other physics engines can be included. Other supported libraries are OpenAL for three-dimensional sound effects, Distributed Interactive Simulation (DIS) and the High Level Architecture (HLA) (Dahmann, 1998) for distributed simulation and CAL3D for character animation. Delta-3D does not only support these libraries but grants also full access to it, therefore we are not restricted by the API of Delta3D. Through its modular design it is also possible to introduce other libraries without much effort.

The Delta-3D engine is not only a 3D rendering engine, but also supports distributed simulation, therefore we choose this platform as the underlying simulation engine to develop the simulator. The Nvidia PhysX Engine allows us to run the dynamics simulation on the graphics card. Nvidia however is closed-source. Another alternative is Bullet which also has a hardware acceleration support. The simulator can be divided in dynamics simulations, controller simulations and environment simulations using the HLA interface for distributed simulation. More about distributed simulation will be described in the next section, where we describe the functionality of the Symbricator3D simulation environment.

2.4.2 The Symbricator3D Simulation Environment

Based on the Delta-3D game and simulation API, we are developing Symbricator3D, a simulation tool especially for modular and swarm robots (Winkler & Wörn, 2009). Symbricator3D is written in C++ and consists of three main modules, robot actors, robot controllers and simulation components. The Delta-3D GameManager is handling all these modules and is responsible for the communication between them. While the robot actors are the physical and geometrical representation of the robot, the controller class represents the software that will run on the real robot as well. Simulation components are responsible for the control of the simulation, e.g. there can be components for user interaction, components for supporting artificial evolution and for setting up various experiments or for simulating the environment.

The robot actors in Symbricator3D consists of four different elements: bodies, actuators, connectors and sensors. Fig. 2.28 shows the configuration of the robot model in the simulation based on the first prototype of the SYMBRION/REPLICATOR robots.

Bodies describe the geometrical and physical properties of parts of the robot like for example the collision shape and the mass of a wheel. A body is a single unit, there are no parts inside a body that are movable. Actuators are those components

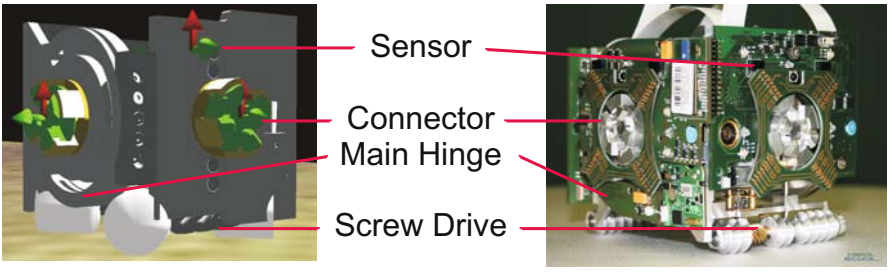


Fig. 2.28 Current robot design in simulation compared with the first prototype.

that connect two bodies together and that can put a force between these two bodies so that they move relatively to each other. Actuators are realized by using joints provided by the dynamics engine. Connector modules for connecting two robots together and sensors modules for perceiving the environment are attached to a body. During the initialization phase the elements are created and put together according to the given robot design. This way, not only different robots can be created in the simulation, but also tools can be developed. Tools are objects which are not fully autonomous robots, but give the organism additional sensors like a high resolution camera or additional actuators, such as wheels to make the organism move faster in flat terrain, or other abilities like a charging unit which enables the organism to recharge its robots at a common power socket. The same way robots can dock to each other, they can also dock to these tools.

2.4.2.1 Sensor Simulation

Several sensors are implemented in the Symbricator3D simulation environment. These sensors can be divided in two different classes: internal and external sensors. Internal sensors describe properties of the robot, such as the position of the actuator or the position and orientation of the robot in a global coordinate system. These values can be retrieved from the dynamics simulation or from Delta-3D. Table 2.7 lists the sensors that are currently supported in Symbricator3D.

External sensors are those sensors that measure the environment. The data for external sensors are retrieved from several sources. Visual sensors such as a camera or a color sensor are modeled by using the camera model of the Open Scene Graph (OSG) library. Based on OpenGL, OSG builds a scene graph where we can define several cameras, which we use for different types of sensors. With the OSG cameras we cannot only model cameras and color sensors but also distance or light sensors, as we can retrieve different scene data from the GPU including luminescence and the depth component of the camera.

The second class of sensors is ray trace based, which is needed for simulating some types of sensors accurately. For a very detailed simulation of infrared or sound sensors for example the reflection at objects is important. This cannot be handled by OSG or OpenGL completely. For such a simulation several rays need to be created

that intersect with other objects in the scene. At the point of intersection a new ray will be created which has the direction of the resulting reflected ray. This way sensor behavior near objects, in corners and even the interference with other sensors can be analyzed. However, such a sensor needs very much computational power and therefore only a few can be simulated at the same time. For an overall simulation of the whole organism this sensor simulation would take too much time. In that case the sensor needs to be approximated by other methods, for example by the OSG camera model.

The third class of sensors is that which needs an additional environmental model. This is not supported by the libraries mentioned above. Temperature, humidity and the distribution of gases for example fall into this category. For those environmental properties a model needs to be developed that describes and simulates their environmental behavior. The modeling, simulation and data storing for the environmental property needs to be handled by a simulation component. The model of the distribution of a gas describes the diffusion and evaporation rate. This needs to be simulated taking the environment into account (for example it cannot diffuse through walls). Additionally it can interact with other environmental properties like temperature or air current which are simulated by other simulation components. For each property there is exactly one component which simulates it (using the Singleton software pattern). The sensor accesses this model through the Game-Manager and receives the current value according to its position and orientation (or velocity or acceleration if required).

A very important point in sensor simulation is the development of an accurate sensor model, which also includes simulation of several types of deviations, like noise, drift and sampling rate. Some sensors are also to some extend sensitive to properties other than the property they measure. For example, most sensors are influenced by the temperature of their environment. As it is possible that the robot can become very warm, such behavior needs to be taken into account when modeling the sensor. To keep the computational time of the CPU to a minimum when simulating the sensor, we need also consider simulating parts of the sensor on the graphics card (GPU). Camera noise for example can be simulated by using shaders. In this case we could render the camera image including noise on the GPU while using the CPU otherwise.

2.4.2.2 Actuator Simulation

To simulate the dynamics of the SYMBRION/REPLICATOR robots, three different actuator models are implemented. There is the main actuator, also called the 3D-drive that enables organism movements and the chain and the screw drive for moving the robot in single mode. Currently two different robot designs (Kernbach *et al.*, 2008a) are implemented. One design uses a chain drive and enables the robot to move over small obstacles, while the other one uses a nearly holonomic screw drive that enables the robot to be more flexible in building organisms. Each of these designs includes the main actuator.

Table 2.7 List of Sensors implemented in Symbricator3D and the underlying library that have been used to implement them.

Sensor Type	Supporting/ Underlying Library	Description
Infrared	OSG	using the depth component of an OSG camera the closest point in the camera image is chosen and the distance to the sensor is returned
Luminance	OSG	the image of the luminance component of an OSG camera is returned
Microphone	OpenAL	OpenAL provides the scene with 3D sound. However OpenAL currently only supports one recorder (the microphone sensor)
Docking Sensor	OSG / Delta-3D	ray trace based sensor that detects a LED of another robot to support the docking mechanism
Laser Scanner	OSG / Delta-3D	ray trace based laser scanner
Fast Laser Scanner	OSG	simulated laser scanner using the depth component of a OSG camera
Camera	OSG	returning an image based on the RGB component of a OSG camera
Color Sensor	OSG	returning the mean color value of a OSG camera (RGB component)
Light Sensor	OSG	returning the mean brightness value
Angular Position	ODE	angular position of a ODE joint
Angular Velocity	ODE	angular velocity of a ODE joint
Force and Torque Sensor	ODE	accumulated force and torque vector which an actuator is currently applying to the robot body parts
Accelerometer	ODE	returns the linear and angular acceleration vector of the robot body parts
GPS	Delta-3D	returns the global position and orientation of the robot

The 3D-drive can be easily implemented with a hinge joint that is provided by the dynamics simulation engine ODE. Hereby, two bodies are connected through a hinge. The hinge constrains the movement of these two bodies relatively to each other, i.e. the bodies can only have a rotatory motion along one axis. According to these constrains the dynamics engine calculates the corresponding constraining forces, a force that moves the bodies in such a way that all constraints are fulfilled. Additional constraint angles – the range in which the actuator can operate – and constraints for the desired velocity of the joint and the maximum force the motor can apply to achieve that velocity can be introduced to that joint. With these constraints

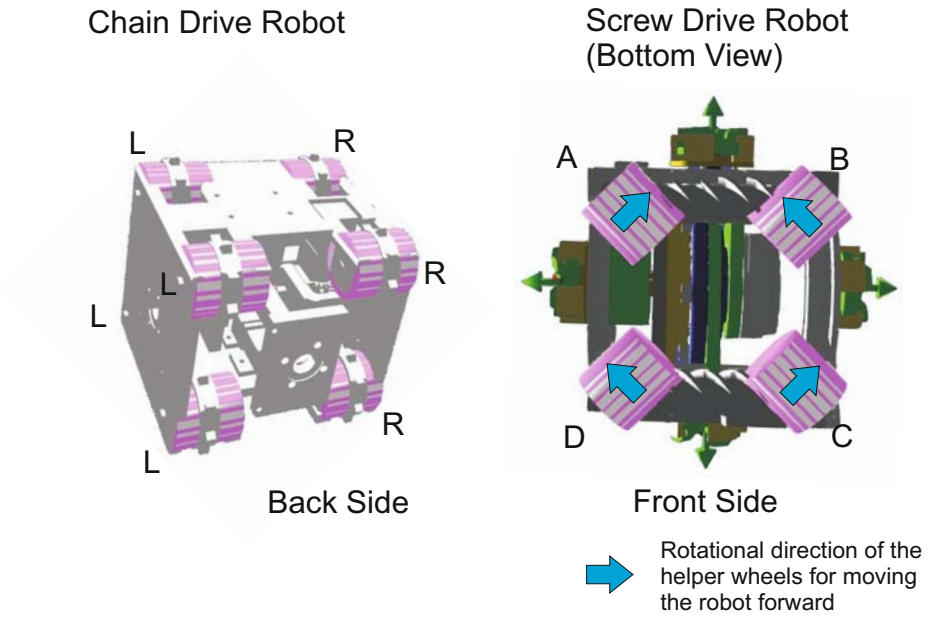


Fig. 2.29 Simulation model of chain and screw drive. The chain drive is simulated by four right wheels("R") and four left wheels ("L"). The screw drive is simulated by four wheels ("A","B","C","D").

major parts of the actuator can be modeled. For getting information about the current state of the actuator we can use the feedback functions of the dynamics engine (see also 2.7). The dynamics engines provide the force and torque vectors the joint is applying to the two body parts of a robot, the current position of the joint and its velocity. To model a sensor monitoring one of these values, deviations of this sensor like described for sensors need also be taken into account.

The other two actuators – the chain drive and the screw drive – are mainly used in swarm mode, i.e. when the robot is decoupled from the organism, but can also be used within the organism with restrictions. As simulating these two actuators in detail would consume too much computational power, the actuators will be approximated through helper wheels. These helper wheels consist of a geometrical cylinder object that is attached through a hinge joint to the main body of the robot. The configuration for the both robots are depicted in Fig. 2.29. The chain drive behaves like a differential drive. If both chains are moved in the same direction with the same velocity the robot is moving in the same direction with the same velocity (if no slip is existent). If moving the chains in opposite directions with the same velocity the robot rotates at the spot. In Symbricator3D the chain drive is simulated by four helper wheels attached at each corner of the chain and are controlled like the real chain. In that way the simulated robot has the same behavior like the chain-driven one.

Table 2.8 Types of Locomotion of the Screw Drive, how they can be accomplished and how they will be simulated.

Type of Locomotion	Front Screw	Back Screw	A	B	C	D
Translation:						
Forward	+1	+1	+1	+1	+1	+1
Backward	-1	-1	-1	-1	-1	-1
Right	+1	-1	+1	-1	+1	-1
Left	-1	+1	-1	+1	-1	+1
Rotation:						
Right around Front Screw	0	-1	+1	-1	0	0
Left around Front Screw	0	+1	-1	+1	0	0
Right around Back Screw	-1	0	0	0	-1	+1
Left around Back Screw	+1	0	0	0	+1	-1

Simulating the screw drive is more complicated, as depending on the usage of the screws the locomotion behavior changes. To move the screw-driven robot forward or backward, the screws need to rotate in the same direction. In that case the screws are behaving like two rollers. If the robot needs to move sidewards, the screws need to be controlled in opposite directions with the same velocity. In that case and if the floor is penetrable the robot is screwing the floor which leads to a sidewards locomotion. If we have a hard floor that is not penetrable like a stone floor, the mechanism does not work and the robot cannot move sidewards. The same holds for the rotation. If the screws cannot penetrate the floor the robot cannot rotate. For rotation one of the screws needs to stand still while the other one is rotating. In that case the robot is moving around the axis that stands still. According to these basic conditions we can formulate the behavior for a pure rotational or pure translatory motion. To finally simulate the screw drive we use four helper wheels that are attached to the robot as depicted in Fig. 2.29. With this configuration we can generate a locomotion in every direction and can generate every rotation. According to the locomotion model of a pure rotational and pure translatory motion of the robot we can calculate how we need to control the helper wheels to achieve such a locomotion. In table 2.8 the different locomotions are listed and how to control the helper wheels to achieve these locomotions.

To simulate a combined locomotion of rotation and translation, we first divide the screw motions into a rotational part ω_1 and ω_2 of the two screws which will lead to a pure rotatory motion of the robot with

$$\omega_1 = 0,\ \omega_2 = sign(\omega_2) \times (|omega_2| - |omega_1|) \text{ if } |s_1| < |s_2| \text{ and}$$
$$\omega_2 = 0,\ \omega_1 = sign(\omega_1) \times (|omega_1| - |omega_2|) \text{ otherwise}$$

and a translational part v_1 and v_2 which would lead to a pure translatory motion of the robot with

$$|v_1| = |v_2| \ .$$

and the actual velocity of the screws s_1 and s_2:

$$s_1 = \omega_1 + v_1$$
$$s_2 = \omega_2 + v_2$$

After that we calculate the ratio of rotation and translation and control the helper wheels according the proportion of the pure rotatory and pure translatory motion.

This model approximates the behavior of the screw drive but does not represent the screw drive exactly, as reliable hardware of the screw drive is not available yet a more detailed model could not be developed. For the next version of the screw drive simulation we will measure the forces and torques the screws are applying on the real robot for different velocities of the screws and according to these values we will develop a more precise actuator model. This new screw-drive model will provide us with the forces the screws will apply to the robot. We can use these forces to put them directly to the physical robot model (i.e. its body parts). The helper wheels will be only passive allowing roll friction for the simulated screw-drive locomotion.

Additionally to all actuator models we implement deviation parameters occurring in the actuator, including noise, short term and long term drifts. In this model variations between actuators of the same type and the resulting different behavior needs to be taken into consideration as well. Therefore we will use on the one hand the data sheets of the actuator manufacturer, but on the other hand we will also make experiments to figure out the behavior of the actuator, so that we can develop very detailed actuator models.

2.4.2.3 Connectors

Connectors are the physical and graphical representation of the coupling device of the SYMBRION/REPLICATOR robots in the simulation. The coupling device gives the robots the ability to connect to each other to build a larger robot organism, which can fulfill tasks the single robot cannot fulfill. Additionally, the coupling device does not only enable the robots to communicate with each other over a internal bus system (Ethernet) it also gives the robots the possibility to share energy and computational power within the organism.

If two robots want to connect to each other, two conditions need to be fulfilled: Firstly, both robots need to be ready for a connection and need to want to connect. Secondly, both robots need to be in the right position and need to have the right orientation. The coupling device has a little margin within the robots are able to connect. In the connecting phase the robots will be aligned by the coupling device so that they fit exactly together. Once connected the robots are immovable to each other. The connectors in simulation are modeled in the same way. There are three tolerance variables that can be set. One for the maximum distance the robot can have to be still able to connect and two maximum angle variables: the twist angle and the pitch angle tolerance. If the two connectors are within these tolerance ranges they can

start the connecting phase. In this phase, forces will be applied on both connectors that will move the two robots the connectors are attached to, to the exact position. If this position is reached, the robots will be connected together by a joint provided by the dynamics engine. After completing the connecting phase the simulated robots are immovable to each other.

The connectors in simulation like the coupling devices on the real robot also provide the robots with communication inside the organism. Two kinds of communication are available within the organism, local communication between two directly connected robots and communication within the whole organism.

2.4.2.4 Communication

Modular robots are often equipped with a variety of communication devices. For example, the SYMBRION/REPLICATOR robots have infrared sensors with which they cannot only detect obstacles, but can also communicate with other robots in their vicinity. Additionally they will be equipped with a ZigBee module for global communication. Within the organism they will have the ability to exchange information with all the other robots in the organism via an internal bus system, and they can send messages to those robots directly connected to them.

Symbricator3D supports also those communication types, whereas the communication is realized through simulation messages, which consists of the message content and the message type. The data type of the message content can be defined by the controller programmer. The message type can be one of the four possible communication types:

Global	Every robot in the simulation receives the message.
Infrared	As the infrared sensor will also be used for communication, this sensor will be used to determine the robots that are in its vicinity. These robots are receiving the message.
Organism	Every robot in the organism receives the message.
Local	A direct neighbor in the organism receives the message. The message will be send through the connector and the recipient is therefor defined.

Symbricator3D checks which robot controller can get the message according to communication type and accessibility, and then sends the message automatically to the right robots. For example, if a robot sends an infrared message, only the robots (and those controllers) within the range of the IR emitter will receive the message.

2.4.2.5 The Controller Class

While the robot class is the physical and geometrical representation of the robot, the controllers are the software representation. They are designed to run in the simulation as well as on the microcontroller without any changes in the source code. This will be achieved through a hardware abstraction layer (HAL), which hides the underlying hardware from the application software, so that the application software

(i.e. the controller) does not need to be changed to run on different platforms. The benefit is firstly the ability to change the underlying robot hardware without affecting the application and secondly the possibility to run these applications in simulation as well. Currently, there are three different operating system on which the application need to run. Firstly, applications can run on the Cortex microcontroller. The underlying operating system is in this case FreeRTOS, a portable open source mini Real Time Kernel. Secondly, application can also run on the Blackfin microcontroller, the second type of microcontroller with which the robot will be equipped. On this microcontroller a μC-Linux kernel is running as operating system. And thirdly, applications run in simulation as well. The simulation can run on different operating systems, like Linux or Windows. As rewriting the application for each platform is too expensive, we are developing the SymbricatorOS (Szymanski *et al.*, 2009b) which will be the HAL to all the applications and for every platform.

Depending on the underlying hardware the applications the type of multitasking processes differs. On the Cortex microcontroller the applications run as FreeRTOS threads, while on the Blackfin controller and in simulation they run as POSIX threads. To benefit of the full performance of the Blackfin processor it requires running the applications as tasks. On the application layer this will not be visible, as the SymbricatorOS is hiding the type of process as well as the different hardware the robot might be equipped with. This includes the simulation as well in such a way that the applications cannot tell by the type of access to the sensor and actuators whether they are simulated or are coming from a real robot. To achieve this a controller base class has been developed which has access to all simulated sensors, actuators and other accessories (e.g. LEDs or batteries). The SymbricatorOS will run in the simulation as a controller which gains access through the controller base class. The applications on the other hand will run on the SymbricatorOS and therefore they will always have the same interface to the underlying hardware, no matter whether it is simulated or not. As on the real robot, in simulation one robot can be controlled by several applications, but each application can only control one robot directly. If several applications are controlling the same robot they need to communicate with each other. On the robot this will be achieved through inter-process communication while in simulation we will have a message-based communication between applications. SymbricatorOS will provide for both mechanisms a uniform interface so that the type of communication between two applications will not be visible to the applications.

Evaluating an application only in simulation might lead to errors. In simulation the application might have much more computational power for example as it will have on the microcontroller. For some microcontrollers there exist emulators which can simulate the behavior of the microcontroller, but as the emulator needs to communicate with the simulation through inter-process communication which requires additional CPU time and running several emulators at the same time would also slow down the simulation, we did not investigate this option further. A much easier way to test applications thoroughly is to run them on the microcontroller itself. The application runs therefore on an evaluation board and is communicating through a serial interface with Symbricator3D, which simulates the environment and within the

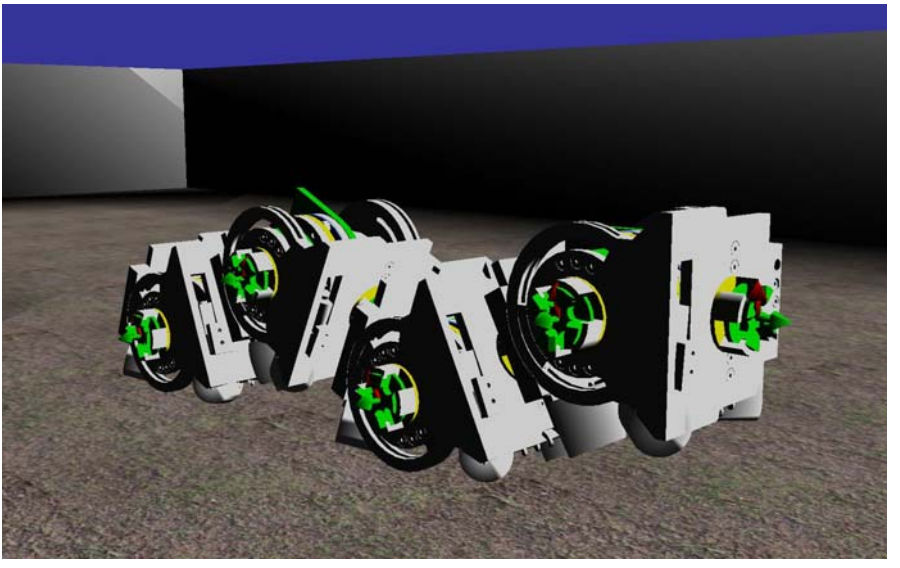


Fig. 2.30 Screen shot of four robots which are connected to a small caterpillar and are controlled by four programs running on four different Cortex evaluation boards which are connected to the simulation.

robots including their sensors. The sensor values are being sent every time step to the evaluation board. The applications on the board are evaluating these sensor values, calculating how to control the actuators and sending the actuator values back to the simulator. In Fig. 2.30 a screen shot is taken of four robots building a caterpillar organism, where each is controlled by a *MDL2ε* program (Szymanski & Wörn, 2007) running on four different Cortex evaluation boards. Through coordinated movements the organism is executing a caterpillar-like movement.

2.4.2.6 Simulation Components

Simulation components are responsible for controlling and observing the simulation and its elements. They can be used for logging robot behaviors, evaluating controllers, resetting the simulation or even simulating environmental variables or other phenomena. As there are lots of different requirements to the simulator, there will be also a lot of different components. Following is a list of simulation components without making claims to be complete:

1. *Dynamics Simulator.* Currently we use ODE for simulating the dynamics in Symbricator3D, which is integrated in the Delta-3D environment. Other dynamics engines however will be investigated in the future to evaluate which one will be the most suitable for simulating modular robots. Three dynamics simulation are currently coming into the closer choice. Nvidia PhysX is a very powerful dynamics simulator which also supports hardware acceleration. Running as a

CUDA application on the GPU the CPU will be released and will be available for other tasks. However, it needs to be shown that the Nvidia PhysX engine is stable and fast enough for our requirements. As Nvidia PhysX is closed source, other dynamic engines which use the graphics processor for calculating the dynamics can be a better choice as it might be necessary to adopt the simulation engine to the requirements of the modular robots simulation. Such a dynamics simulation is Bullet, which uses shader programming on the GPU for this task. Another very promising approach for simulating the dynamics is the Impulse-Based-Dynamics-Simulation (Bender, 2007)(IBDS), which is faster and more stable compared to the widespread Lagrange Multiplier(LM) method (Schmitt & Bender, 2005).

2. *Kinematics.* At the end of the project we aim to have around one hundred robots which are connected together. Simulating an organism with so many degrees of freedom entirely dynamically might not be suitable as it can lead to instabilities. To overcome this problem a solution is to switch off the dynamic simulation of hinges that are not currently used and simulate them only kinematically. A kinematics component will take the task and thus support the dynamics component.
3. *2D-Simulator.* In swarm mode robots are not docked together and do not use their hinge for locomotion. In this case, simulating robots dynamically is not always necessary. To speed up simulation, the robots will be simulated by a 2D simulator which approximates their locomotion with a simplified locomotion model.
4. *Simulation Decision Maker.* The different simulators mentioned above will be running at the same time, each responsible for simulating different parts of the scene. During simulation runtime it is necessary to switch the robot simulation between these simulators. Therefore, a component needs to decide how each robot need to be simulated and assign them to the appropriate simulator.
5. *Environmental Simulator.* Different environmental phenomena which are not already provided by other libraries but which can be measured by the robot need also be simulated. Such phenomena are for example the dispersion of gases, temperature or humidity. Their behavior is simulated by environmental simulation components. If a robot is equipped with such a sensor, the simulated sensors will access the corresponding simulation component and will retrieve the current value according to its position and orientation or if necessary to its velocity or its acceleration.
6. *Arena.* One of the major aspects within the SYMBRION/REPLICATOR projects is to develop new bio-inspired algorithms. To test and evaluate these paradigms, whether it is online or off-line evolution or other kinds of machine learning, the simulator needs to provide mechanisms to measure the quality of these algorithms. Therefore, robot behaviors need to be logged, controllers need to be evaluated and diverse fitness functions need to be calculated. The arena component is offering functions for controlling, supervising and evaluating the robot behaviors and therefore it will help to evaluate the controllers. The arena component offers among others the controlling of the simulation, including pausing, resetting and loading new configuration into the simulation as well as observing and logging robot behaviors.

2.4.2.7 Distributed Simulation

Controllers and simulation components are not only implemented thread-safe so that they can run on other cores but they are also implemented in such a way that they can run distributed on other computer platforms. This way the simulation can be easily distributed over several computers connected via an Ethernet network. With this mechanism we can even distribute the simulation not only locally but also through the Internet. For distributed simulation we use the High Level Architecture(HLA) standard which allows to combine several simulations into one simulation (Dahmann, 1998). Each simulation, in HLA terms called federate interacts with other federates through a Runtime-Infrastructure(RTI). The RTI is the connection between each simulation. In Symbricator3D each simulation component can run on a different computer platform, as well as the robot controllers. To reduce the traffic controllers and their robots should run on the same platform. The dynamic simulation of an organism however cannot be distributed and need to run on one platform. Each computer platform is chosen in such a way that they fit their task the best.

For example, a highly GPU powered computer runs the dynamic simulation of the scene including all the physical robot models. If robots or other objects do not need to be simulated dynamically, a kinematic or a fast 2D agent simulator takes the task. Several processors simulate sensor and controller behavior and additionally some environmental simulations can be used to simulate phenomena such as temperature or humidity dispersion and diffusion. Each federate can run distributed. The RTI makes sure that the information needed by other federates will be provided in time and that the federates are synchronized regularly. Fig. 2.31 displays exemplary how a distributed Symbricator3D simulation can look like.

The distributed simulation also offers the possibility to visualize sensor values of the real robots or organisms. In this case the robot/organism runs a small federate that is sending the sensor data through a W-LAN module (attached at a robot or implemented as a tool). The simulation uses the sensor data and visualizes them for the user. For example the data of the infrared sensors of all the robots in the organism

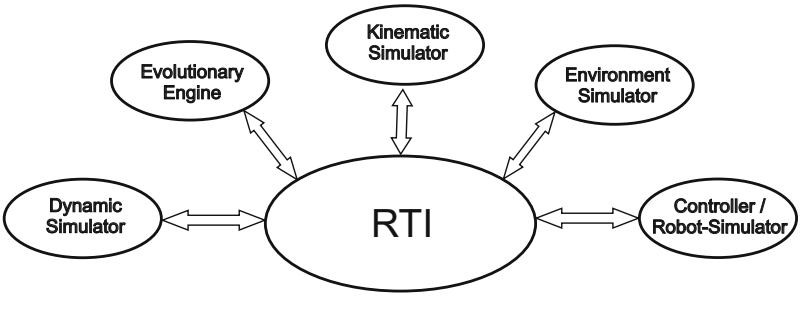


Fig. 2.31 HLA - Distributed Simulation: Several Federates (Simulations) are simulating different phenomena for Symbricator3D and communicating over the Runtime Interface (RTI).

are gathered together and are sent to the simulation, which uses the data to visualize the surrounding obstacles. Using the simulation the user can also control the robots/organism remotely, whereas his commands will be sent back to the organism. As the wireless LAN connection is fast enough even camera images can be broadcasted to the simulation. This way debugging algorithms on the robot will become easier. Additionally through this mechanism an organism can source additional computational power out to a more powerful workstation.

2.4.3 Showcase: The Dynamics Predictor

To show how Symbricator3D can be used in each stage of developing a controller program, we like to introduce the idea of the dynamics predictor. The aim of the dynamics predictor is to compute how each joint needs to be controlled so that a given movement pattern can be obtained. A movement pattern is a recurrent sequence of actuator movements to move the organism. This can be a walking pattern or a pattern for rotating the organism for example. For each type of organism, there exist different types of movement patterns. To evaluate the quality of an organism, the best movement patterns need to be known. Evaluating each possible configuration would take an enormous amount of time as the possibilities are growing exponential with the number of robots in the organism. While there are four different possibilities for a two-robot-organism in 2D, there are already eight different possibilities for an three-robot-organism. 2D means in this case that the robots are only rotated around the Z-axis.

2.4.3.1 L-Systems

As the search space is enormous and the time needed to evaluate an organism, it is essential to develop an intelligent, heuristic search strategy to find suitable organism forms. One approach is to define families of organism forms whereas in an organism family there are all these organisms with the same applicable movement patterns. Caterpillars with different size for example can be controlled with the same algorithm, hence they are in the same family. Before describing how to control robots in a family with the same algorithm we need to describe organism families. One promising approach are the L-forms, which have been introduced by the Hungarian theoretical biologist and botanist Aristid Lindenmayer in 1968 to model the growth process of plants (Rozenberg & Salomaa, 1980) and which have also been used to develop and simulate artificial creatures in computer animation (Hornby & Pollack, 2001). L-systems are defined as a tuple:

$$G = \{V, S, \omega, P\}$$

where

V is a set of variables
S is a set of constants
ω the starting sequence
P a set of production rules

Using his L-system, Lindenmeier described the growth of algae as follows:

$V : A, B$
$\omega : A$
$P : (A :\rightarrow AB), (B :\rightarrow A)$

which produces:

$t = 0 : A$
$t = 1 : AB$
$t = 2 : ABA$
$t = 3 : ABAAB$
$t = 4 : ABAABABA$
$t = 5 : ABAABABAABAAB$

At the first step, at $t = 0$ only the first rule $A \rightarrow AB$ can be applied and we get the term AB. In the next step we have to use both rules once. While A becomes AB again B becomes A and we get the term ABA.

Hornby and Pollak (Hornby & Pollack, 2001) have used parametric L-Systems to produce families of structures to create different creatures in simulation. In parametric L-Systems production rules can also have parameters and algebraic expressions can be applied as parameter values for the successors. In the next example are the following production rules P given:

$a(n) : ((n > 1) \wedge (n \bmod 3 \neq 0)) \rightarrow Aa(n-1)$
$a(n) : ((n > 1) \wedge (n \bmod 3 = 0)) \rightarrow Ba(n-1)$
$a(n) : (n \leq 1) \rightarrow A$

With the starting sequence $\omega = a(6)$ the following string will produced:

$t = 0 : a(6)$
$t = 1 : Ba(5)$
$t = 2 : BAa(4)$
$t = 3 : BAAa(3)$
$t = 4 : BAABa(2)$
$t = 5 : BAABA(1)$
$t = 6 : BAABAA$

As with the parametric L-System any kind of configuration can be described, we will use the L-system to model the morphology of our organisms.

2.4.3.2 Describing the Structure of a SYMBRION/REPLICATOR Organism Using L-Systems

To use the parametric L-systems for creating different families of organism forms, we need to define where and how a robot can connect to the organism. We define the following variables and constants:

$V : X_i'$
$S : X_i, A, B, C, D, :, [,]$

with X_i the robot in the organism, A, B, C, D the sides of the robot, ":" the representation of a connection between two robots, "[" the operator which saves the current position in the organism and "]" the operator which restores that position. X_i describes a robot in the organism while X_i' describes a position where a robot still needs to dock to the organism.

Using this grammar we can describe every 2D organism. If we want to describe a 3D organism we need to extend the constants for the sides A, B, C, D with an index indicating how the robot is docked to the organism. The docking device is 90° symmetric, meaning the robot can be rotated by $n \times 90°, n \in \mathbb{N}$ around the coupling device axis and it is still possible to be docked to the organism (in case the robot will be rolled to the side, the robot will loose its ability to locomote alone). To indicate that a robot is docked in such a way, we'll add an index n to the side constants, e.g. A_1 means the robot is docked with its A-side, rotated by 90°.

With this grammar we can describe the organism either by using identifiers for each robot or by classifying them. In the first case X_i represents a robot with the unique identifier i. In the later case X_i defines a class of robot. This class defines different robot types in case of a heterogeneous organism and the functionality of the robot at that position (e.g. the spine or a leg in the organism). If we have an organism consisting only of homogeneous robots, we can even resign the identifier i completely. A mechanism to figure out the structure of an organism without using identifiers has already been achieved with the CONRO robots (Salemi & Shen, 2004). Based on hormone-inspired messages, a protocol has been developed that enables each robot module to discover its location and correct type within the organism. With little modifications the mechanism can be adopted for the SYMBRION/REPLICATOR robots.

If X_1 and X_2 are two robots that are connected to each other, whereas X_1 is connected with the side H_0^1 and X_2 is connected with the side H_0^2 we describe it as:

$$X_1[H_0^1 : H_0^2 X_2] \; with \; H_0^1, H_0^2 \in \{A, B, C, D\}$$

With this L-System representation we can describe every possible connection between two robots and therefore we can describe every organism. In Table 2.9 a few examples of different organism types are given including their L-System representation and their production rules for the organism family. Within this list there are two special cases. The first one is the wheel. For this organism it is necessary to give one robot an identifier (in this case it is also the initializing robot X_0), so that

another module in this organism knows where to dock. With this mechanism, we can create and display every circular dependency in an organism. The second case is the alternated caterpillar, where one robot has been rotated around its connector axis. As this robot cannot rotate itself by its own, it needs first be rotated by an organism (It can be the same organism or a different organism which is specialized for this task). Afterwards the organism needs to dock to the robot, as the robot cannot move anymore. In the L-System representation as well as in the production rules, it is easily visible when such an action needs to be executed before further building up the organism. In the given example the third robot needs to be rotated, as it is described in the grammar, indicated by the index of the side variable ($\ldots X_1[A:C_1X_2\ \ldots$). The forth robot on the other hand has been rotated back, also indicated by the index of the side variable ($\ldots X_2[A:C_3X_1\ \ldots$) and therefore can connect to the organism as usual. However not every configuration, that has been created by the generator can be achieved. Therefore a validation step is necessary.

2.4.3.3 Validating the String Representation

An organism representation for the SYMBRION/REPLICATOR robots is valid when the following requirements are fulfilled:

1. Maximum one robot can dock at each side. There are four sides.
2. If X_i and X_j in a L-System representation s represent the same robot, a valid joint angle representation $q_1,\ldots,q_n$ needs to exist so that $X_i = X_j$
3. If X_i and X_j in s represent two different robots X_i and X_j, a valid joint angle configuration $q_1,\ldots,q_n$ needs to exist so that X_i does not collide with X_j and the second postulation is still valid with this joint angle configuration.

The first point can be easily assured by validating the string s. that has been generated by the L-System. The string s is valid when it has the following structure:

$$s = X_0[A:s_1]^{0..1}[B:s_2]^{0..1}[C:s_3]^{0..1}[D:s_4]^{0..1}$$

with s_i a substring with the following structure:

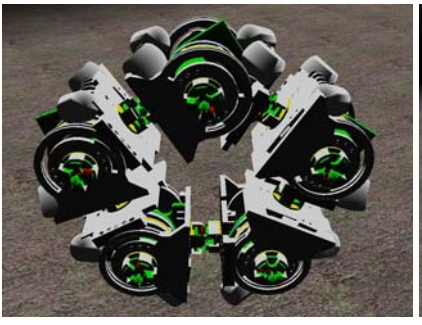
$$s_i = H_0^i X_i[H_1^i:s_j]^{0..1}[H_2^i:s_k]^{0..1}[H_3^i:s_l]^{0..1}$$

with H_k^i the sides of the robot X_i, whereas $H_k^i \neq H_l^i,\quad k \neq l,\quad k,l \in \{0,1,2,3\}$ and s_j, s_k, s_l are substrings with the same structure as s_i. The expression $[...]^{0..1}$ means that the expression $[...]$ can occur not once or once at this position in the string. According to these restrictions to the string representation, we can define the structure of production rules. The production rules for the initializing robot X_0 and the other robots X_i need to have the following structure:

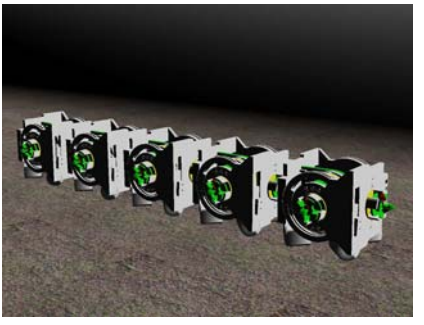
$$X_0' :\rightarrow X_0[A:H_0^1X_1']^{0..1}[B:H_0^2X_2']^{0..1}[C:H_0^3X_3']^{0..1}[D:H_0^4X_4']^{0..1}$$

$$X_i' :\rightarrow X_i[H_1^i:H_0^jX_j']^{0..1}[H_2^i:H_0^kX_k']^{0..1}[H_3^i:H_0^lX_l']^{0..1}$$

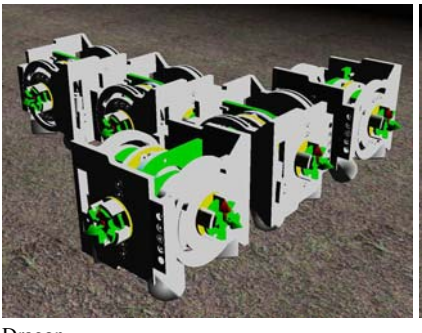
Table 2.9 Organisms, their L-System Representation and their Production Rules.



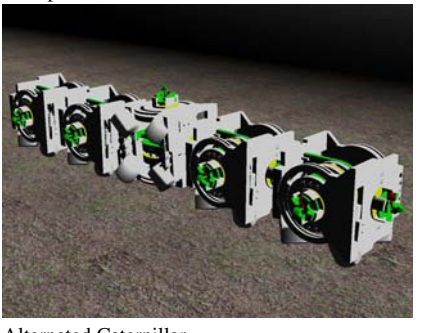
Wheel



Caterpillar



Dragon



Alternated Caterpillar

Organism:	Wheel
L-System Representation:	$X_0[A:CX[A:CX[A:CX[A:CX[A:CX_0]]]]]$
Production Rules:	$X_0'(n) :\rightarrow X_0[A:CX'(n-1)]$
	$X'(n):(n>1)\rightarrow X[A:CX'(n-1)]$
	$X'(n):(n\leq 1)\rightarrow X_0$

Organism:	Dragon
L-System Representation:	$X_0[A:CX[A:CX]][B:CX][D:CX]$
Production Rules:	$X_0'(n) :\rightarrow X_0[A:CX'(n-1)][B:CX'(n-2)][D:CX'(n-2)]$
	$X'(n):(n>1)\rightarrow X[A:CX'(n-1)]$
	$X'(n):(n\leq 1)\rightarrow X$

Organism:	Caterpillar
L-System Representation:	$X_0[A:CX[A:CX[A:CX[A:CX]]]]$
Production Rules:	$X_0'(n) :\rightarrow X_0[A:CX'(n-1)]$
	$X'(n):(n>1)\rightarrow X[A:CX'(n-1)]$
	$X'(n):(n\leq 1)\rightarrow X$

Organism:	Alternated Caterpillar
L-System Representation:	$X_0[A:CX_1[A:C_1X_2[A:C_3X_1[A:CX_1]]]]$
Production Rules:	$X_0'(n) :\rightarrow X_0[A:CX'(n-1)]$
	$X'(n):((n>1)\wedge(n \bmod 3=0))\rightarrow X_2[A:C_3X'(n-1)]$
	$X'(n):((n>1)\wedge(n \bmod 3=1))\rightarrow X_1[A:C_1X'(n-1)]$
	$X'(n):((n>1)\wedge(n \bmod 3=2))\rightarrow X_1[A:CX'(n-1)]$
	$X'(n):(n\leq 1)\rightarrow X_1$

Regarding these rules when defining the production rules the initializing robot can have maximum four connections to other robots and at each side only one robot can be docked. The same holds for the rest of the robots. They are docked to their parent robot at one side and maximum three other robots can be docked to them. This way the first point has been assured. The other two points however cannot be so easily gained, without restricting several organism forms that might be possible.

The second two points can only be proven by the inverse kinematics. Using the L-System representation, the kinematic chain of the organism can be developed. Therefore the representation string s needs to be transformed into several equations which describe the transformation between the robots in the organism.

2.4.3.4 Kinematic Representation

A kinematic description of the robot is essential for the organism, as it helps to find out the relative position and orientation of other robots or tools. With this knowledge sensor fusion of gyroscopes in different robots for example will become possible. With the inverse kinematic it is also possible to move a robot or a tool to a given position to allow docking with other organisms or to plug into a power socket. The previously described L-System representation of the organism can be used to develop a kinematic representation of the organism. Therefore each robot X_i is a homogeneous matrix that describes the position and orientation of that robot relatively to another robot or to the world coordinate system. The sides of the robot A, B, C and D can also be understood as homogeneous matrices, which describe the transformation of the docking elements relatively to the center of the robot. In Fig. 2.32 the assembly of the two robots is displayed. The side A and D are in both designs fixed to each other, while the other two sides are connected to AD via a hinge in the one design and via two different hinges in the other design. We define A and D as independent to the hinge, i.e. the orientation of A and D define the orientation of the robot. The orientation of B and C are joint dependent, with $B(0)$ and $C(0)$ the initial state of the two sides. The matrices of the sides are defined as:

$$A = \begin{pmatrix} 1 & 0 & 0 & -d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad B(0) = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$C(0) = \begin{pmatrix} -1 & 0 & 0 & d \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

As the robot and its coupling devices are symmetric to 90° rotations there are four possibilities how two coupling devices can connect to each other. These rotation is described by the roll matrix R_n:

$$R_n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos n' & -\sin n' & 0 \\ 0 & \sin n' & \cos n' & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad with\ n' = \frac{\pi}{2}\, n,\ n \in \{0,1,2,3\}$$

B and C are also joint dependent and to calculate their final orientation a pitch matrix $P(q)$ is necessary:

$$P(q) = \begin{pmatrix} \cos q & 0 & -\sin q & 0 \\ 0 & 1 & 0 & 0 \\ \sin q & 0 & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Using these transformation matrices, we can finally describe the exact transformation of each side of the robot:

$$\begin{aligned} A_n &= R_n\, A \\ B_n(q_B) &= P(q_B)\, R_n\, B(0) \\ C_n(q_C) &= P(q_C)\, R_n\, C(0) \\ D_n &= R_n\, D \end{aligned}$$

whereas q_B is the joint angle of the actuator at the B-side and q_C the joint angle at the C-side. There are two different SYMBRION/REPLICATOR robots. One has one degree of freedom, the other two. In the case of only one degree of freedom $q_B = q_C$.

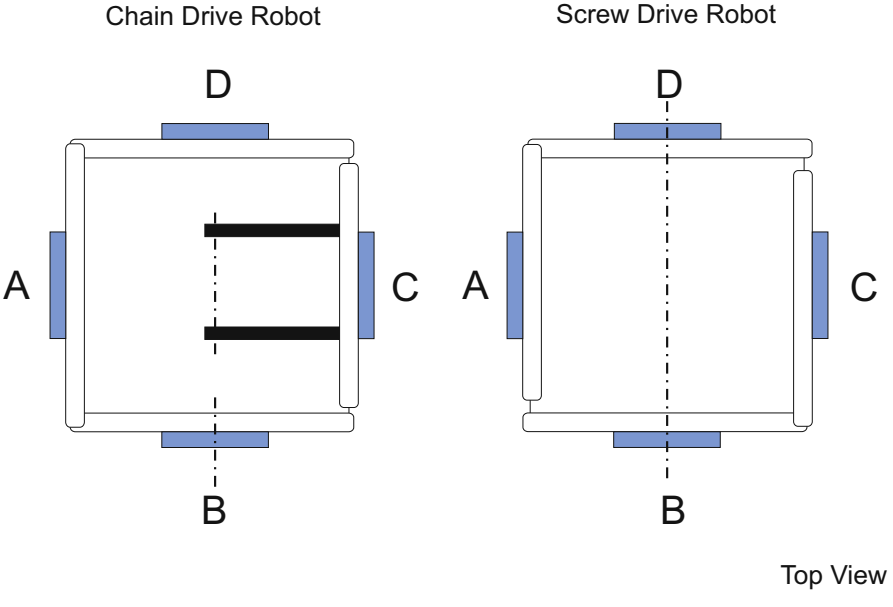


Fig. 2.32 Rotatory Axes of the Chain Drive Robot and the Screw Drive Robot.

To obtain an equation that describes the transformation of robot X_i to another robot X_j, we need to transform the L-System representation into an equation. In a valid representation string s, we will find a substring in the form of $X_i[H_i^j : H_j^i X_j]$ (the side H_i^j of robot X_i is connected to side H_j^i of robot X_j). As this substring defines the connection between robot X_1 and X_2, we can describe the kinematic transformation between these two robots with the equation:

$$H_i^j \, X_i = R \, H_j^i \, X_j \tag{2.7}$$

with R the rotation matrix of 180° around the z-axis (The two robots are rotated by 180° towards each other when docked):

$$R = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Referring to Eq. 2.7 we introduce the transformation matrix T_j^i which transforms the coordinate system of robot X_i to X_j so that $X_i = T_j^i X_j$ applies:

$$T_j^i = \left(H_i^j\right)^{-1} R \, H_j^i$$

whereas H_j^i and H_i^j are one of the four homogeneous matrices listed in Sect. 2.4.3.4. Let X_0 and X_n be two arbitrary robots in the organism and $X_1, ..., X_{n-1}$ robots in the organism, whereas X_1 is connected to X_0 with its side H_1^0 (and X_0 is connected to X_0 with its side H_0^1), X_2 is connected to X_1 with its side H_2^1 and so on, then the equation

$$X_0 = T_1^0 \, X_1 = T_1^0 \, T_2^1 \, X_2 = T_1^0 \, T_2^1 \, T_3^2 \, \ldots \, T_n^{n-1} X_n = T_n^0 \, X_n$$

applies, with the transformation matrix T_n^0

$$T_n^0 = \left(H_0^1\right)^{-1} R \, H_1^0 \, \left(H_1^2\right)^{-1} R \, H_2^1 \, \ldots \, \left(H_{n-1}^n\right)^{-1} R \, H_n^{n-1}$$

In case of one degree of freedom per robot, T_n^0 depends on maximum $m \leq n+1$ joint angles, with two degrees of freedom it is a maximum of $m \leq 2 \times (n+1)$ joint angles.

Given the kinematic chain as described above, we can now solve the forward kinematic problem, meaning we can determine the position and orientation of a robot module relative to another robot module in terms of the joint variables. Especially, we can determine the position and orientation of an end-effector, a tool for example. The inverse kinematics problem is to find a suitable joint configuration so that the end-effector or any other part of the organism is at a desired position with the right orientation. This is very important, as it is necessary for docking two robot organism, docking an organism to a tool or to interact with the environment like plugging the organism to a power socket. To solve the inverse

kinematic problem a joint angle configuration $q_0, \ldots, q_m$ needs to be found so that the equation $X_0 = T_n^0(q_0, \ldots, q_m)\, X_n$ applies. The inverse kinematic problem can be solved numerically or analytically. Numerical methods are computationally intensive. They can become unstable or can oszillate. Different numerical methods have been implemented by Kelmar and Khosla (Kelmar & Khosla, 1988) and Höpfler and Ottler (Hopler & Mosterman, 2001; Hopler & Otter, 2001) for example. Analytical methods on the other hand are exact and not as computational intensive. However, for a kinematic redundant robot, there does not exist a closed-form solution. The same holds for a robot with six degrees of freedom in general. As the geometry of the organism is unknown, a general solution for the inverse kinematics would result in a numerical solution. For reasons of stability and computational intensity however it should be aimed for an analytic solution for most of the organism structures, which means that the organism geometry needs to fulfill particular requirements (Hollerbach, 1984; Manseur & Doty, 1992a; Manseur & Doty, 1992b; Tourassis & Ang, 1995).

2.4.3.5 Example: Caterpillar

In the following, we will explain the parametric L-System description for the SYMBRION/REPLICATOR organism and the kinematic equation along at hand of the caterpillar organism. Later we will use this example to further extend the L-System grammar to also describe the dynamic behavior of the organism and to introduce movement patterns. The caterpillar form for example is one of the simplest organism families, which is able to overcome small obstacles and gaps in the environment.

It can be described with the following production rules:

$$
\begin{aligned}
&X_0'(n) : && \rightarrow X_0[C : AX'(n-1)],\\
&X'(n) : (n > 1) && \rightarrow X[C : AX'(n-1)],\\
&X'(n) : (n \leq 1) && \rightarrow X,
\end{aligned}
$$

which initialized with $\omega : X_0'(4)$ produces:

$$
\begin{aligned}
&t = 0\text{: } X_0'(4)\\
&t = 1\text{: } X_0[C : AX'(3)]\\
&t = 2\text{: } X_0[C : AX[C : AX'(2)]]\\
&t = 3\text{: } X_0[C : AX[C : AX[C : AX'(1)]]]\\
&t = 4\text{: } X_0[C : AX[C : AX[C : AX]]]
\end{aligned}
$$

In this case we have only one type of robot, as each robot in the caterpillar has the same function to fulfill. We have also no identifier declared, as we do not need it in this example. The only robot, that slightly differs from the others in the organism is the X_0 robot. This robot initializes the organism and will be responsible for generating impulses that will trigger an organism movement. According to the production rules all robots know their location in the organism and if other robots need to dock

to them. Additionally they also know their function. The scenario for the caterpillar organism is the following.

At the beginning X_0 is in swam mode and encounters a problem that it cannot solve alone, like climbing a wall. The robot needs to classify the problem and needs to decide which organism type is suitable for that kind of problem. Let's assume the robot decides to create a caterpillar with four elements. It sends locally (via infrared) a message to other robots, that it wants to create an organism. Another robot gets this message and moves towards X_0. The new robot now receives from X_0 further information about type of organism, where to dock with which side to dock and how many robots are left (parameter n) and its function (X). The new docked robot will now look for further robots. This will be done as long as robots are missing in the organism or a timeout aborts the procedure. The timeout is essential as we do not want the robots to wait infinitely. If there are not enough robots to build the organism, necessary to fulfill the task, the robots need to detach from the organism again and need to look for more robots. This is another challenging part in the project.

We assume we have enough robots and the last robot just docked to the organism. This robot will broadcast a message that the organism has been completed and robot X_0 begins to send reflex messages periodically. The behavior of the caterpillar and its robot modules has the following structure:

1. The robot X_0 begins moving its hinge up- and downwards using a wave function and after Δt seconds it sends a reflex signal to its direct neighbor. It keeps sending this signal with the period of the wave function.
2. All the other robots are beginning the same motion when receiving the reflex signal. After the same Δt they are forwarding that signal to the next robot.

This way the organism is executing a wave function defined by the particular wave function of the modules and the velocity of propagation $c = l/\Delta t$ is defined by the robot length l and Δt the time between receiving a reflex and forwarding it to the next robot.

2.4.3.6 Movement Patterns

Based on the L-system that defines the organism structure, we can now easily develop such movement patterns. A movement pattern is a set of rules for each module type in the organism, whereas each robot is deciding what to do, according to its function (i.e. the module type) in the organism and its sensor and communication input. There is no central mechanism to control the organism. The movement patterns can for example be described in the motion description language $MDL2\varepsilon$. For our example, the caterpillar organism, we need to define the movement pattern of the initializing robot X_0 and the other robots in the organism, which are in this case all of the same type. As both behaviors only differ in creating or forwarding the reflexes, the behavior can be described in one plan.

Listing 2.1 *MDL2ε* plan for a caterpillar movement

```
<PLAN name="CaterpillarMovement" duration="infinite">

  <BEHAVIOUR name="noLocalMessageReceived" interrupt="OR(NOT(
      ILOCAL_MESSAGE_LIST_EMPTY), EQ(ITYPE, 0))">
    <ATOM name="AROTATE_HINGE" interrupt="ITRUE" duration="1" arg0="-0.5"/>
    <ATOM name="ASEND_LOCAL_MESSAGE" interrupt="ITRUE" duration="1" arg0="1"/>
    <ATOM name="AROTATE_HINGE" interrupt="ITRUE" duration="1" arg0="0"/>
    <ATOM name="AROTATE_HINGE" interrupt="ITRUE" duration="1" arg0="0.5"/>
    <ATOM name="AROTATE_HINGE" interrupt="ITRUE" duration="1" arg0="0"/>
    <ATOM name="ACLEAR_MESSAGE_LIST" interrupt="ITRUE"/>
  </BEHAVIOUR>

</PLAN>
```

In this plan two different elements are appearing, the ATOM and the BEHAVIOUR. An ATOM describes what to do, e.g. moving the hinge, sending a message or switching the red LED on. An ATOM will be executed as long as the boolean interrupt expression is true and the time given in duration has not been elapsed. Additionally arguments can be given to the atom, like moving the hinge to the angular position -0.5 radians. The BEHAVIOUR is in this case an aggregation of ATOMs, but it can also include additional BEHAVIOURs or other *MDL2ε* elements. A complete description of *MDL2ε* can be found in (Szymanski & Wörn, 2007).

At the beginning all robots are waiting to receive a local message (i.e. a message from a direct neighbor), except robot X_0 ($ITYPE = 0$) which starts to move its hinge at first to position $arg0 = -0.5$ for one time step. Afterwards it is sending a local message. The robot next to it starts the same movement after receiving the message and so on. This way all the robots begin their movement time-delayed by one time step. At the end of the movement the robots are clearing their message list, so that they do not enter the BEHAVIOUR element again. Through this mechanism all the robots are synchronized in their movement and a caterpillar movement establishes. Fig. 2.33 shows that movement using this plan.

Fig. 2.33 Caterpillar Movement: Several robots are docked in a row and by only moving the hinges of each robot like a caterpillar the organism moves forward.

2.4.3.7 Evolving the Organism

MDL2ε has already been successfully used to evolve different behaviors for differential - driven swarm robots, beginning with obstacle avoidance and wall-following to more complex patterns like escaping a labyrinth (Szymanski *et al.*, 2009a). As our example is not much more complex as a wall following pattern, *MDL2ε* can also be used to evolve these movement patterns. However, evolving more complex behaviors will take a few days even with a network of state-of-the-art computers. To

evolve movement patterns for every possible organism configuration would take too much time. Therefore, movement patterns for organism families need to be evolved. As simulating small organisms is less time consuming, evolving movement patterns for them is faster. Therefore we aim the following strategy:

1. Create an organism family using the parametric L-System.
2. Evolve movement patterns for the first member of the family (Create the organism with the smallest parameter possible).
3. After evolving the movement, evaluate the fitness of the organism family.
4. If the fitness is high enough, it is allowed to grow. That means, generate a new member of that family with more modules.

Only organism families with the best fitness are allowed to grow and are evaluated again. While growing the organism another interesting aspect would be to additionally change the production rules. This way, it gives the possibility to first generate a caterpillar, which is later be able to evolve legs for example. After growing the organism, the best movement patterns of the previous organism will be used as a starting point for evolving movement patterns for the new one. This way, the organism can become very complex and we will still be able to evolve movement patterns as we rely on previous gained knowledge.

As not every organism is suitable for every task, the organism needs to change its phenotype during runtime. For example, the organism has found a suitable structure for moving over a gap, but then needs to climb a stair. It is forced to change its phenotype. In simulation we can easily establish different scenarios and different fitness functions to establish an organism library where for each situation a suitable organism structure will be saved. This library will be saved on the real robot afterwards and if the organism encounters a new situation it access the library and chooses a new – for the new situation suitable – form. The movement patterns are also saved within the library, which the organism can access.

2.4.3.8 Dynamic Representation

Like the kinematic representation describes the geometrical relations between the robots in an organism, the dynamic representation describes the physical relations between these robots and therefore the motion of the organism. When for example a robot in the organism moves its main actuator, the main actuator applies a torque to both parts of the robot. These torques will be forwarded to the next robots. At the same time the torques of the main actuators of the other robots are also applied to the bodies of that robot. As we know, the geometrical structure of the robot and its physical properties we can use the L-System representation to model the dynamics of the organism. The knowledge about the dynamics gives the possibility to control the actuators more precisely. To achieve a given movement pattern, we will develop a dynamics predictor which will calculate the required torques for each robot in the organism. It will also be able to constrain robot velocities in the organism and therefore can prevent these modules from damage.

The dynamics predictor will react to divergences and external events that interfere in the desired movement that need to be achieved. Based on sensor data from the main actuator, it will calculate the current positions, velocity and acceleration of each actuator and it will detect with touch sensors which robots lie on the floor. Additionally, gyroscopes will provide information about the orientation towards Earth and the acceleration of the module and air pressure sensors will give information about the elevation of each robot module. These sensor data will be fused together and in a second step it will be fused with sensor data of other robots, so that we get information about position and orientation of the whole organism, as well as its kinematic and dynamic behavior. Using this information the dynamics predictor will calculate which forces need to be applied on each joint to achieve the position of each joint for the next step. The positions of each joint (and therefore each module) will be provided by the movement pattern. As divergences – either through simulation errors or disturbances from outside – need to be compensated, actuators need to be able to apply higher forces than in simulation. It is therefore essential to have margins in simulations when evolving movement patterns, so that the organism in reality is able to perform the movement patterns.

To calculate how to control the actuators, the dynamic predictor needs to calculate the inverse dynamics of the organism. This calculation however is very computationally intensive and cannot be solved by one micro processor in time, but as each robot module in the organism has a powerful micro processor which is connected by an internal bus the organism itself possesses a distributed computer network. The larger the organism the more computational power will be available, but the more complex will be the calculation of the inverse dynamics as well. To be scalable the dynamics predictor needs to have a memory and a calculation complexity of not more than O(n). Additionally the calculation needs to be easily distributed through the organism. As a base for the dynamics predictor we will take the impulse-based dynamics simulation IBDS (Bender, 2007), which we already introduced in Sect. 2.4.2. It is not only a suitable dynamics simulator for the Symbricator3D simulation, its impulse based approach also fulfills all the requirements for the dynamics predictor.

In (Schmitt & Bender, 2005) Bender compared his impulse based simulation with different implementations of the Lagrange multiplier (LM) method according to stability and accuracy. The Lagrange multiplier method is well-known and widely used, amongst others in the Open Dynamics Engine. He simulated a single, double and a triple pendulum and compared constraint, energy and oscillation time drift. He came to the concussion that applications using the LM method become unstable and inaccurate if not fully stabilized, while the impulse-based methods characterize a high stability. Additionally, the impulse-based dynamics simulation has a computational and memory complexity of O(n). Therefore and because the algorithm can be distributed to multiple cores, it cannot only be integrated in a network of powerful micro processors such as we have in our organism, but it is also scalable towards simulating dynamics for larger organisms.

2.4.4 Conclusion and Future Work

In this section, we presented the Symbricator3d simulation environment to simulate modular robots such as the SYMBRION/REPLICATOR robots that are currently being developed in the these European projects. The simulator is designed in such a way that it can simulate modular robot organisms as well as robot swarms. Additionally different kinds of sensors and actuators are implemented. As the computational requirements are too high to be fulfilled by one computer the simulator is designed to be distributed over a network. As there are different simulations involved which need to interact with each other, the distributed simulation standard HLA has been chosen. This way simulations for dynamic behaviour, robots and their controllers and the environment can be developed independently and communicate through a uniform interface. To illustrate each stage of the controller development using the simulator, we presented the L-System to represent the structure of the organism and we used movement patterns and a dynamics predictor to achieve the corresponding organism motions. Using these algorithms, different movement patterns will be developed in the simulator and later, when reliable hardware is available, will be evaluated on the real robots.

The simulator will also be extended to be able to simulate robots and the organisms in different ways depending on the current requirements, e.g. robot swarms

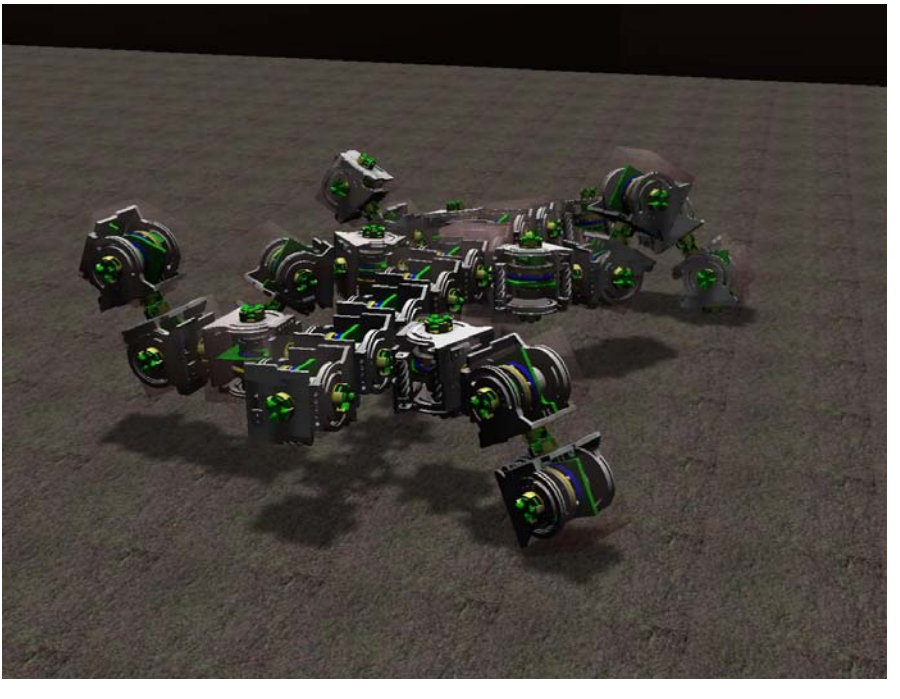


Fig. 2.34 Several robots that are connected to a six legged organism. The transparent boxes around the robots display their current collision model.

will be simulated with a fast 2D simulator and organisms with a dynamics simulator. When the organisms become too large to be simulated completely dynamically a kinematic simulation will support the dynamics engine in such a way that not all joints are simulated dynamically anymore. Additionally we will have different collision models for the robots which can be exchanged during runtime. In Fig. 2.34 for example the robots in the organism are approximated by boxes to reduce the computing time for the dynamics simulation. In other cases such as the docking maneuver of two robots a very detailed collision model of the connectors is mandatory. Depending on the current requirements the simulator will decide which collision model needs to be used.

The different motion simulators (for robot swarms and organisms), as well as environment and robot-controller simulators will be distributed using the HLA standard. The interface therefore will be developed and implemented into the Symbricator3D environment in the next months. As the real robots will be equipped with wireless communication devices such as ZigBee or W-Lan modules it will even be possible to connect these robots to the simulator. The simulator will be able to display the robots and their senor data and therefore will offer a good debugging and observation tool for developing modular robot controllers.

Chapter 3
Cognitive Approach in Artificial Organisms

3.1 Cognitive World Modeling

Libor Přeučil, Petr Štěpán, Tomáš Krajník, Karel Košnar,
Anne van Rossum, Alfons Salden

The chapter introduces possibilities and principal approaches to knowledge gathering, preprocessing and keeping in autonomous mobile robots' artificial organisms. These may comprise "classical AI" concepts as well as "new AI principles", whereas both approaches themselves may bring up either major advantages, or suffer from certain drawbacks.

The classical approach relying on sensor-fusion-model-planning and actuation schema takes the advantage of explicit representation of the organism knowledge which may be represented by varied types of world model structure (Barrera, 2005). Subsequently, these structures are mainly understood as geometric and other environmental features carriers. Features and data are considered for explicit representation of world properties and typically have precisely known location, meaning and confidence. These properties serve for inputs to cognitive or planning subsystems allowing to execute reasoning processes over this data. Major advantages of this stand in predictable behaviors, strong data reduction ratio, and therefore better possibility of tracking and safety of the robot operation.

The disadvantage of this class of methods remains in a stiff way of combining specific cognitive methods, typically capable of adjustment to slowly changing organism operating conditions. Therefore, the process of adaptation/evolution to rapidly changing environment condition becomes a hard problem in this approach.

The other approach aiming to store knowledge in an implicit form tends to be more flexible in adaptation/learning and evolution aspects and ranges from Brooks' principles to Neural Net knowledge representations. Hence, this advantage is balanced by unknown or fuzzy localization and form of particular knowledge. Estimation of particular behaviors and possibility of their determination remains low. Moreover, due to undetermined meanings of particular knowledge/data components, efficient filtration of data amounts becomes ineffective.

P. Levi and S. Kernbach (Eds.): Symbiotic Multi-Robot Organisms, COSMOS 7, pp. 165–228.
springerlink.com

The leading target in here stands in elaboration of novel approaches to representation of world knowledge based of combination of selected features of the above mentioned methods. A hybrid approach, that combines strong data amount reductions and easy understanding of a World Map content with high flexibility of the "new AI" principles is proposed.

3.1.1 Methodology

The world model captures knowledge the robot has about its surrounding environment. The aim is to store world information and world-robot interaction in a common structure allowing efficient distribution, merging and combination of various knowledge.

The world can be described amongst a connectionist versus symbolist axis, as well as a spatial or non-spatial axis. Is there any way to reconciliate those different views on the world model? An ad-hoc combination of those different strategies on a robot platform is not desired. A fruitful combination of the modules developed from the different angles, can be found in the multi-agent system (MAS) (Wooldridge, 2002) methodology.

Conventionally, the agents in MASs are considered as entities that are rule-based or algorithm-based. There is however no reason to extend this to more adaptive control architectures, such as neural networks. Conventionally, agents in MAS research are static. There is also with respect to this no reason that there are no composition operators defined on the agents. Multiple agents might be merged to single agents and agent-boundaries do not necessarily have to be predefined.

In the field of neurodynamics, Kozma (Kozma, 2008) implements a multi-agent system built out of so-called multi-scale Kachalsky-sets or K-sets. A hierarchy of K-sets originating from an individual level of inhibitory and excitatory cells leads from limit cycle oscillations, to chaotic oscillations towards spatio-temporal dynamics with global phase transitions.

A modular approach is also taken by Edelman in the field of neural darwinism, which field is currently exploding. For example, read the work of Maniadakis and Trahanias (Maniadakis & Trahanias, 2006). Their supposed model consists out a collection of neural agents, each one representing a brain area. They argue for the use of evolution because implementing architectures with hundreds of modules might be unfeasible (Popescu-Belis, 1997).

The above mentioned examples are heavily skewed to neural implementations. There is however no reason to only build neural agents, but a wide diversity of machine learning mechanisms might actually be applied (Drogoul & Zucker, 1998).

3.1.2 Spatial World Modeling

As a first-sketch approach may be assumed insertion of known representations of world and world-robot interaction knowledge between the sensor subsystem and

the reasoning and evolutionary engine of the robot. This improves observation possibilities of the system performance and strongly reduces sensor data flows from the sensors via execution of data fusion techniques. Suitable data fusion methods to achieve this are presented herein.

The next step elaborates on possibilities and approaches to distribution of these data representations. These are needed to handle knowledge organization in both the organism and the swarm modes and outline necessary procedures during transition between these two modes.

Another foreseen dimension of abstraction to be used is found in diverse granularities of the contained data representations. The basic investigated world abstraction includes the geometric and simple symbolic representations. The latter ones allow to come up with topological models of the environment. As a result, this enables to design multi-leveled environment model (map) comprising the topological, geometrical and procedural knowledge representation levels at once. The knowledge stored in a topological map node is constrained only by limitation of hardware and it can describe any property of the environment including its dynamics.

The aforementioned approaches are tested and studied with respect to their embodiment into low-power/low performance hardware to enable real-world testing of the design principles in laboratory experiments.

Therefore, certain sensing technologies will be preferred. This will include a smart camera system integrated with structured light-based ranging device, various proximity sensors, inertial units for organism body determination in space, RF based localization system etc.

3.1.3 Evolution Map

The Evolution Map (EMa) is a modular framework dedicated to provide information suitable for evolution, navigation, localization and spatial reasoning. Moreover, the EMa has been suited also for action planning and to support human-robot interaction and knowledge exchange. Methods incorporated in the EMa framework design build together a unified approach to the environment and the robot-environment interaction representation. As the robot-environment interaction patterns depend on robot body configuration, the robot morphodynamics can be modeled as well.

The map in this framework is rather understood as a mean, or a process, incorporated into the robot control system, than considered for a fixed result created before. In these terms it may be considered as a knowledge base providing particular information for an intelligent acting, i.e. the representing database contains the procedural (“how”) as well as the declarative (“what”) knowledge. This forms the EMa content, which mainly stores information necessary to know “how to get from one place to another” and/or “how to distinguish one place from another”. Moreover, the stored information can be easily converted into human-oriented and understandable form and can be used in communication with a human.

The EMa modular architecture is open to incorporation of new methods into the framework and therefore open for use with evolutionary techniques. Different

behaviors and algorithms contributing to robot performance can be switched on and off during run time and even cooperate together to achieve improved robustness and reliability. Nevertheless, individual modules do not need to be completely "aware" of each other, their coordination is handled by EMa itself. All the used algorithms, behaviors and their coordination and cooperation are executed by an Executor as well as a Reasoning module.

Consequently, EMa framework consists of the following modules:

- The Map holds gathered information about the environment and provides interfaces for data storage, modification and retrieval, i.e. the map works as a knowledge base.
- The Reasoning module is ordering the data in the map. The module is responsible for insertion of vertices and arcs into the map as well as it is capable to generate the most proper hypothesis consistent with the map content and explaining the given set of observation. The generated hypotheses are subsequently verified with assistance of the Executor. This step stands in suggestion of recommended actions to support verification, rejection or adoption of the hypothesis in question.
- Jockeys are modules that are responsible for sensing and conducting various actions in the environment as well as the map keeping, i.e. insertion of data into the map through predefined interfaces.
- The Executor module provides data abstraction and planning on higher level. This module decides selection and launch of various Jockeys as well as ensures data interchange between Jockeys and the map.

This structure hides the implementation details and enables handling all types of knowledge in unified way. It also enables coordination and cooperation of navigating, learning and localizing algorithms.

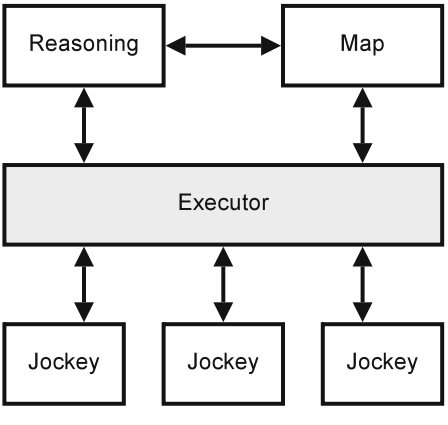


Fig. 3.1 EMa Architecture.

3.1.4 Map

The map is a core part of the framework which holds all information about the environment and the environment-robot interaction. The main structure used to represent the map is a directed graph which consists of vertices and arcs representing known topological relations. In this topological structure, the vertices stand for places in the environment, whereas the arcs represent possible navigation paths. If needed, metric representation of related environment parts (local metric maps) can be attached to each vertex and/or arc as an additional information, see Fig. 3.2.

Moreover, the vertices can also be interpreted as points of interest or significant milestones in the environment. Typically, these are assigned to places that the robot can reliably recognize and even where the robot can undertake certain decisions about its future behavior for planning purposes. Therefore, each vertex needs to carry labeling information possibly distinguishing it from other vertices as well as information used to localize the robot inside the local frame of reference. The labeling information is typically related to certain features of the environment and allows detection of the robot position whenever found in the particular vertex. Nevertheless, the existing vertices do not need to be distinguishable exclusively by their description. A full and unambiguous definition of particular vertex needs both the vertex label and its relation to the rest of the graph.

Each arc stores procedural knowledge, giving a description how to traverse from its initial location denoted by the start vertex to a target place given by the goal vertex. The subsequent behavior of the robot is expected to be deterministic, i.e. the robot placed in the starting position and using certain procedural knowledge navigates always to the same destination. The violation of this expectation is classified for a failure which can be recovered only and only in the navigation phase, but can lead to fatal situations in the exploration phase. To deal with this issue, a probabilistic extension of the traversing behavior may be applied (Ephraim & Merhav, 2002).

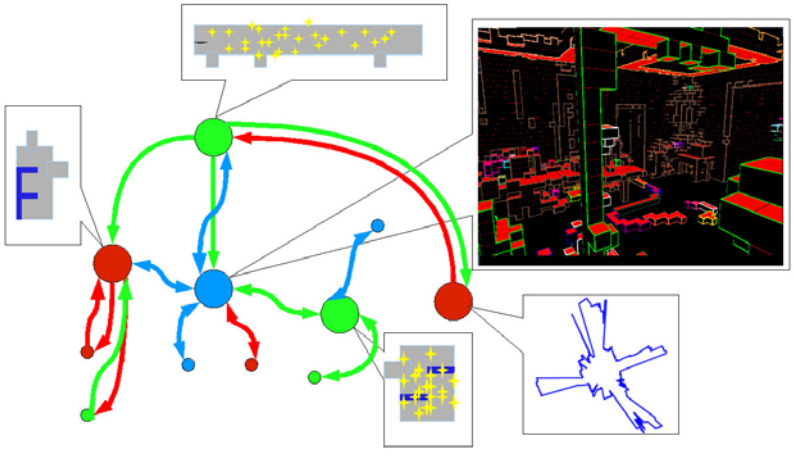


Fig. 3.2 Graphical representation of the map.

As mentioned, every vertex and arc can carry further additional information as metric map of the neighbourhood, its certain features (area, length, curvature, relative position) etc. The additional information and knowledge is intended for further use for planning, reasoning about the environment, and interfacing the map to sensors, actuators and controllers.

Map elements are stored in transaction database-like structure, i.e. every vertex and arc is identified by a unique label. Corresponding arcs then hold identifiers of source and target vertices. If an extension into more detail is required, the basic database elements vertices and edges can be refined by attaching additional information at any level.

Interfaces create abstract layer above the specific type of the data in the map. Each interface takes care about storage of specific information about vertices or arcs. The interface mechanism is transparent so one can request vertices with all available information as well as one can specify the exact desired information. Since the particular information does not affect each other, extension of the existing map with a new class of knowledge without affecting functionality of all other previous algorithms, is trivial. Undesired information is ignored and masked to be invisible to these algorithms. Distribution of the global world model into several submaps bound by topological relations allows keeping these local submaps sufficiently small to reduce a risk of excessive demands on processing power.

3.1.5 Jockeys

Jockeys are modules responsible to perform "riding" of the robot. It means that these modules are dedicated to control the robot actuators and to read the robot sensors in order to achieve certain behaviors. A Jockey has direct access to the robot hardware or can be connected to another module with direct access to the robot hardware. Moreover, the Jockey may be related to one or more map interfaces.

Jockey modules are connected to the core part of the EMa framework through sockets and XML-based protocol. The socket communication allows splitting the executive part from the deliberative part in a logical as well as in a spatial way. The XML protocol itself is easy to be extended and customized, as well as ensures connectivity of different types of control programs and remains independent on the background operating system.

Let us define four different types of Jockeys for the purpose of the EMa framework as:

- Localizing Jockey is responsible for determination of position and orientation of the robot in the world.
- Navigating Jockey is responsible for control the robot along an arc.
- Learning Jockey is responsible for gathering new information about navigation along a robot path.
- Evolution Jockey combines properties of Navigating Jockey and Learning Jockey into a single module.

The Localizing Jockey is related to a specific type of vertices and sensors. Diverse sensors need different Localizing Jockeys for handling specific hardware and processing data, but these Jockeys can utilize the same map interface and therefore they may also share the stored information.

On the other hand, Localizing Jockeys with different algorithms can share the same sensors. If the robot arrives into a vertex, the Jockey in question gathers information about the vertex, computes and stores specific descriptors into the map and determines outgoing edges from the current vertex. Diverse types of Localizing Jockeys can produce different vertex descriptions and also discover different outgoing edges.

A running Localizing Jockey is able to distinguish vertices of the specific type from each other and compare the actual vertex with other vertices stored already in the map. As result, this provides probabilities of being located in some certain and known vertex. In addition, relative positions in these vertices can be estimated. This behavior is used for global localization and loop-closing. A classical examples of Localizing Jockey would be the iterative closest point (ICP) algorithm (Mazl & Preucil, 2000) based on rangefinding methods or a Jockey fusing position from odometry and global localization system by Kalman filtering methods (Thrun *et al.*, 2005b).

The Navigating Jockey takes care for traversing along arcs of a specific type. The Jockey guides the robot along the given edge and determines the end of this edge whenever approaching a next vertex. This Navigating Jockey procedure can have two different outfits: memoryless and memory-based implementations.

A memoryless Jockey relies on a reactive navigating strategy. The algorithm has no a priori information about particular arc except the information needed to distinguish outgoing arcs from others. Main advantage of reactive navigation stands in the ability of traversing along completely unknown edges.

A memory-based Jockey needs to have certain data about the arc in advance. Such information is acquired by the Learning Jockey. Being more informed, the memory-based navigation approach can provide much better performance in terms of higher preciseness, repeatability of actions, etc. than the reactive one. It is important to depict, unless the memory-based approach can reliably recognize a failure to traverse a certain (preexisting) arc while the memoryless Jockey cannot.

Navigating Jockeys can either directly link sensory outputs to actuators, e.g. link camera color recognition to motor actuators to realize path finding or describe complex sensorimotor patterns to realize movement of the organism.

The role of the Learning Jockeys stands in gathering environmental information while robot is driven by a Navigating Jockey. In result, the collected information is stored in a descriptor, valid for the arc the robot has been driven along. The collected data can subsequently be utilized by memory-based Navigating Jockeys. In other words, the Learning Jockey creates a parallel arc with a different navigating strategy to the actually traversed one. Learning Jockeys use the same representation of the environment as the relevant navigating memory-based Jockey. For example, the Learning Jockey can remember color blobs or other image

features (Bay *et al.*, 2008) detected along the way and use these to navigate the learned path later (Matsumoto *et al.*, n.d.).

The Evolution Jockey applies self-organizing techniques to navigate the robot along arcs in order to improve performance of the so far existing Navigating Jockeys. An evolutionary approach is used to improve robustness and efficiency of the robot movements, the list of which may contain: neural networks, homeostasis or hormone system control. Evolutionary activity results into new populations of Jockeys with diverse properties. The Evolution Jockeys can mutually cross-over to modify themselves or to create a completely new Jockey entity.

3.1.6 Reasoning

The map can be understood as a knowledge base containing a set of facts about the environment and robot-to-environment interactions. Having a proper formal tool it becomes possible to infer new facts from this knowledge base and the set of rules that allow interpretation of the database content. The knowledge in the map is not limited to "crisp" facts, but models uncertainty and probability as well. The subjective logic approach (Josang, 2009) is proposed as a suitable formal logic system to handle uncertain and probabilistic knowledge in the EMa framework. It is suitable also for conditional reasoning which is useful for hypothesis generation and verification.

Facts in subjective logic are called opinions. The opinion ω has four components: believe b, disbelieve d, uncertainty u and atomicity a:

$$b + d + u = 1.$$

The first three parameters define position in the opinion space. The atomicity represents a priori knowledge and is useful in evaluating highly uncertain opinions. The used definitions of belief and disbelief follow the Dempster-Shafer theory (Shafer, 1976). The uncertainty stands for lack of evidence for the given proposition. It is something that fills "void" in the absence of the both belief and disbelief terms. A case with zero uncertainty ($u = 0$) is equivalent to traditional probability, i.e. the believe b is a probability, that a fact is true. Therefore, an opinion with $b = 1$ or $d = 1$ is equivalent to a true or false fact in "crisp" binary logic. For example, an opinion α with $u = 1$ is different from opinion β with $b = d = 0.5$. The β would be an opinion about the outcome of a fair coin toss, while α means that there is no knowledge about the coin fairness.

Above classic logic operators like AND, OR, NOT etc., the subjective logic offers further additional operators: discounting and consensus. These two operators are very useful to combine results from two or more algorithms. Given an opinion about correctness of an algorithm A. Then, the discounting operator takes the resulting outcome of this algorithm and adjusts it making-use of the given opinion about the correctness of A.

The consensus operator allows to combine two opinions on the same proposition provided by two different algorithms. Therefore, this operator may be applied for fusion of the results from Localizing Jockeys. As for the Localizing Jockey result can be a value in range $< 0, 1 >$ or $< 0, \infty >$, it becomes necessary to transform this value into the form of an opinion. Desired transformation parameters for the conversion can be acquired using machine learning techniques from training sets of annotated vertices.

Subjective logic also provides mechanism for abductive reasoning. The abduction allows inference of the precondition φ as an explanation for the observed consequence ψ. It is a method of reasoning which finds the hypothesis that would best explain the relevant evidence. Herein, it is possible to apply the abduction to model the map based on a set of observations.

Subjective logic is computationally and memory undiscerning and therefore can easily be implemented on a SYMBRION/REPLICATOR robot.

3.1.7 Executor

The Executor is responsible for the execution of a plan made by the reasoning module. According to the requested action sequence in the plan, the Executor recalls proper Jockeys. If there are multiple Jockeys admissible to fulfill a certain action, the one with the best expected performance is chosen. The Executor supervises the behavior of the executed Jockey on the fly and in the case the Jockey tends to fail, it tries to substitute the requested behavior by executing another Jockey with a similar function.

During the exploration phase, the Executor aims to run Jockeys to collect the largest possible portion of information about the environment. It tries to run all relevant Learning Jockeys during arcs traversing and all available Localizing Jockeys after traversing ends in the target vertex. Both the multiplicities are aimed at gathering as much as possible data to traverse along arcs and to precisely determine position of vertices.

While the robot is learning the arc, it may also appear, that one or more Learning Jockeys need to stop the robot and/or access specific actuators of the robot. The reasons for this event may originate from a need for more sensor data or lack of processing power for sensor data streams. The Executor grants access to hardware for the requesting Learning Jockey only if it is possible to interrupt the currently navigating Jockey. After the reason for interrupt vanishes, the Executor returns control to the navigating Jockey.

Besides, the Executor also performs localization at the newly visited vertex. The localization stands in undertaking a decision if a robot just has entered a new vertex or just revisited an already mapped (known) one. To approach optimal performance, the Executor evaluates time consumption of each Jockey and optimizes gathered information against the required time. The Executor calls Jockeys in order of their efficiency, until the result is convincing enough.

3.1.8 Porting the EMa onto a Robot

The EMa approach has been suggested to serve for one component in a control system for self-evolvable robot. The term "development of robot capabilities" may not cover only changes in the robot behavior but the evolutionary process can also influence the robot organism physical shape as well as configuration of its body parts (cells).

Allowing such a dynamically changing robot raises a question concerning internal organization of the system. Specifically, assignment of functional components (i.e. control methods and implicit capabilities) to physical parts of the robot (control units and actuators) and their interlinking gives subjects to further investigation.

A standard GOFAI type of robot (Barrera, 2005) comprises all the functional and physical components in one body, which remains unchanged. This approach allows keeping the robot internal configuration fixed over time in majority of cases.

On the other hand, evolvable robots change shape and structure over time in the manner that it cannot be unambiguously pre-determined which part of the robot body shall fulfill what functionality and what knowledge it needs. This fact becomes more critical in the cases, which do not allow access to particular robot capability, method, or data/knowledge. Such a situation appears typically in cases, represented by the process of physical reconfiguration of the robot body cells. These cells typically comprise certain actuators, sensors and computational power allowing to implement data processing and decision-making methods, data and knowledge storage, etc. These, whenever detached from the robot body cannot be accessed any more. Further extension of the previous idea leads to a robot built of similar body cells functional components providing very basic capabilities alone, but strengthening their performance when merged together. The latter represents the main driving force of the evolutionary process and takes the advantage of sharing own and using other's resources and knowledge. Since robot body cells are similar, it can easily be estimated, that the resulting robot after (or in) the evolution process will have evenly distributed available resources, i.e. computing power and data/knowledge storing capabilities over the whole body.

Once the data processing units and the memory are the main features to place the overall behavior of the robot into, it appears straightforward that the "integrated" knowledge/experience and high level behaviors are to be spread over the body cells as well.

Due to evolutionary behaviors, the robot is expected to attach new and detach undesired cells. The newly attached cells have either no knowledge or their knowledge is very different from the rest of the robot body. This fact forms a new problem how to align the newcomer cell with the rest of the body in the sense of sharing resources and knowledge in the robot body with the cell and vice versa. This process of cross-breeding pre-learned knowledge belongs to one of the evolutionary mechanisms and enables functional evolution via qualitative improvement (or collection) of specific knowledge, that allows execution of particular new functionalities. In the case of intentional detaching a cell from the robot body, a complementary situation has to be handled. The knowledge contained by this cell has to be either salvaged

(i.e. replicated into another part of the remaining robot) or will be lost. As the former case preserves the knowledge database (the EMa), the latter necessarily has to fix possible inconsistencies in the map as a consequence of losing a certain part of the data.

3.1.9 EMa Care-Taking Procedures

Simplified procedures presented below (see also Fig. 3.4) for resolution of the afore sketched core problems determine backbone solutions in complex cases. Considering a single-cell case, to keep the EMa representation functional over the whole robot body the following services, in terms of extension to the set of Jockeys, have to be applied:

- The Attach Jockey overtakes responsibility for all necessary operations in the case of annexing new body cells. The cell connection procedure itself is invoked and performed by the evolutionary control system running on the robot, but the Jockey is activated whenever physical connection of a newcomer cell is completed and its part of the EMa database becomes activated. The goal of the Attach Jockey is to incorporate the newcomer EMa into the EMa of the rest of the robot body. This means primarily to unify both the sets of knowledge, in particular by unifying the coordinate systems for submaps followed by a fusion procedures of local (Štěpán *et al.*, 2005) and global topological maps (Huang & Beevers, 2005). This process leads to improvements in the resulting robots EMa by execution of a procedure similar to "select and store the best only".
- The Detach Jockey becomes activated whenever a request for disconnection of a cell (or set of cells) appears. Again, as the requirement for the detach operation (under normal operating conditions) originates from the evolutionary control system the Jockey fulfills its role prior to physical disconnecting of the selected

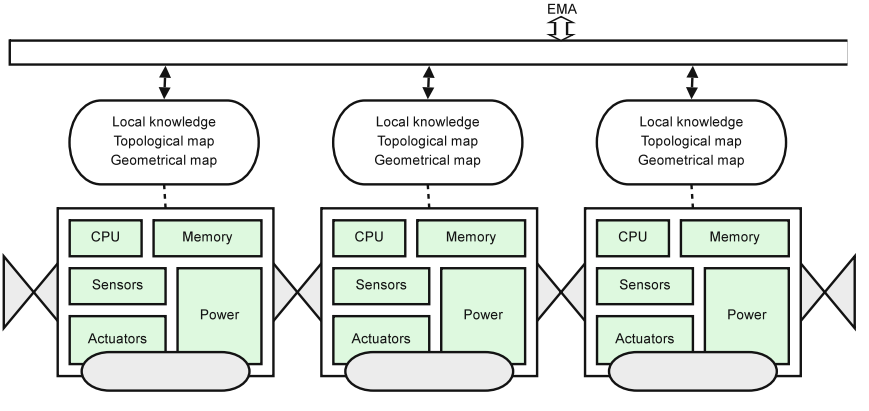


Fig. 3.3 Physical and logical distribution of resources over the robot body.

part from the body and acknowledges this state. Core role of the Jockey is to possibly preserve all the data in the determined cell for the main robot body, i.e. to make a physical reprint of that content somewhere into the remaining part of the robot. Nevertheless, the data salvation process may not always necessarily be successful. Salvation procedure may fail in the cases of insufficient space for data storage in the remaining robot body. As the operation may be executed in both directions, from detaching cell towards the remaining robot body and from the robot body to the detaching part, a single cell is not able to accept all the knowledge from a large body of the robot. This invokes typical situation of data salvation failures from principle. The latter situation can also be interpreted as the larger remains the robot, the more knowledge it may contain and take the advantage of. Once the original robot decides to split into cells or comparable parts, it may be considered for true replication. Flagging successful completion of the Detach Jockey role avoids losing EMa content from the determined cell (or set of cells). The unsuccessful performance of the Detach Jockey falls either into the afore mentioned cases of insufficient space for knowledge replication or denotes a case of forced detach or malfunction of the cell. The latter case can be adjusted by the Recovery Jockey, explained in the following. On the other hand, the insufficient space substantially differs from the forced detach situation as it can preserve consistency of the EMa content. This case may be maintained by choosing only the most informative knowledge to be preserved.

- The Recovery Jockey holds for a self-healing procedure in the cases the body cell detach has not been successfully completed, i.e. EMa consistency cannot be guaranteed. Unless Attach and Detach Jockeys are initiated from above by the evolutionary controller, the Recovery Jockey is invoked always if an inconsistency of EMa content is indicated by an instant background procedure checking for EMa self-consistence. Preconditions for EMa recovery appear exclusively in the forced detach or cell malfunction cases. The role of the Recovery Jockey is to recover consistency (linkage of proper geometric features, mutual references in the database, etc.) in the remaining part of EMa. This healing procedure brings the EMa again back into fully operational shape, nevertheless lacking the lost content of the detached or broken cell. Runtime of the Recovery Jockey is flagged as a "not ready" state of EMa for the rest of the robot.

3.1.10 Physical Layout

Physical implementation of the EMa system is twofold. It is being executed on evenly distributed hardware over the robot body, represented for this case by a set of mutually interconnected CPUs and memory modules physically attached to each particular body cell, see Fig. 3.3. Due to relatively low computational power and memory space available on each cell, stand-alone performance of this setup remains far from the requirements possibly imposed on the system by execution of more sophisticated behaviors than primitive ones (i.e. simple reflex responses, etc.). Therefore, to establish high-level and smart behaviors the evolutionary system is

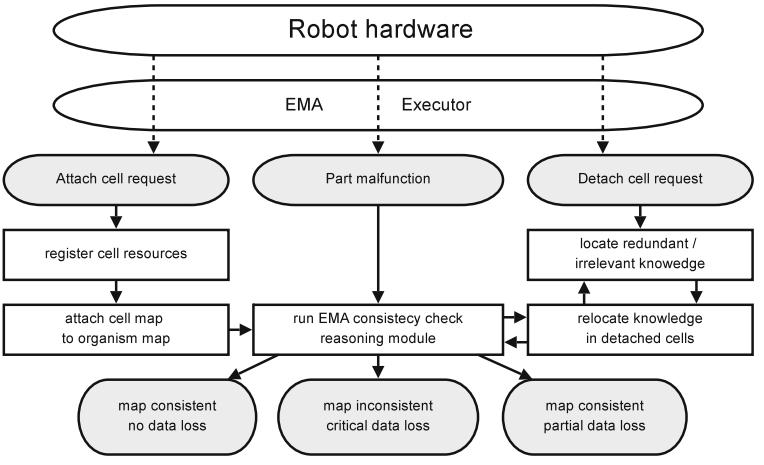


Fig. 3.4 Ensuring consistence of the EMa system.

expected to take the advantage of all available computational and knowledge storage resources onboard the robot organism. The larger the robot appears, the more space for knowledge processing and storage remains available. On the other hand, once the computational/storage system is physically distributed over the robot body, requirements for particular tasks and launching as well as corresponding data sharing needs to be governed by unique authority. This supervising entity keeps track on momentary computational load and usage at certain cells. The overall goal is to achieve optimal distribution and assignment of raising tasks to be processed and data to be stored to ensure the robot functionality as a whole. This functionality is partially be resolved by the used operating system (Szymanski *et al.*, 2009b) on the lowest level. Nevertheless, proper managing of the tasks and memory usage for EMa system remains on an EMa Executor. The main role of the Executor is the internal coordination and handling of data and procedure executions within the EMa framework. All EMa functionalities are exported through a unified interface. That way, the EMa system finally appears from the external view to be a self-healing and self-updating carrier of spatial data (knowledge), with unified interface which is available to other (typically higher level) control procedures running onboard the robot organism.

3.1.11 Logical Layout and Communication

The EMa Executor implementation is approached either in centralized way or in a fully distributed form. As the centralized outfit appears to be the simplest way to achieve the desired performance, it suffers from certain disadvantages. The major weakness of the central implementation is the unique location of the Executor code,

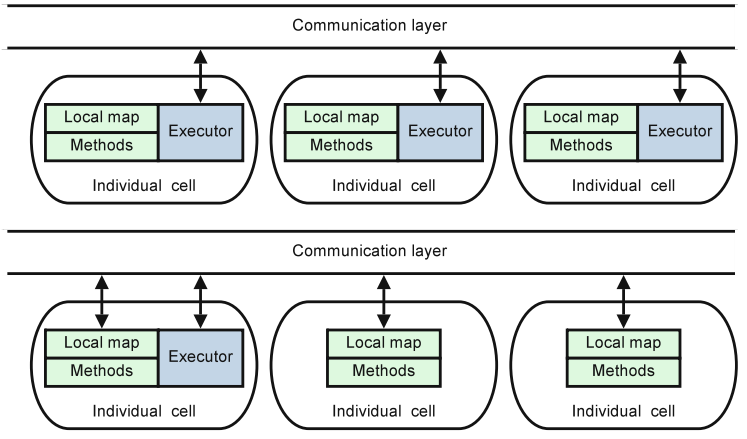


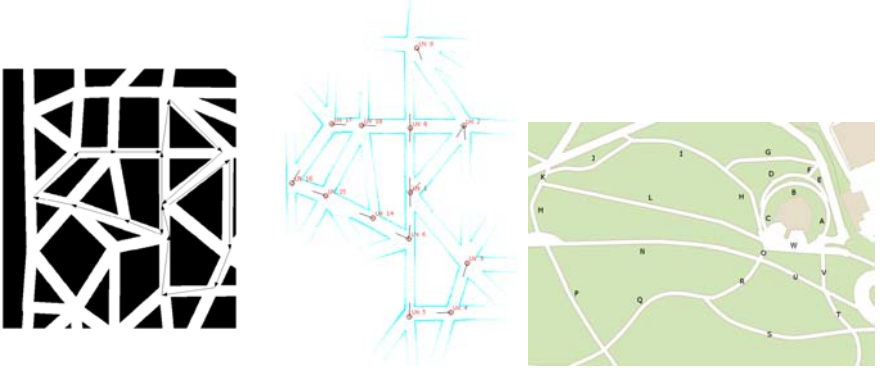
Fig. 3.5 Centralized and distributed approach.

which may be easily affected by occasional malfunction of the physical cell which executes the code. Moreover, unique localization of the Executor code has to rely on functional communication with all the cells it supervises. In other words, once any physical communication link (or a routing cell) fails, all the cells behind this point become inaccessible. This situation may cause malfunction (lost supervision) over a large part of the robot body, the recovery from which may not be easy.

To avoid the afore sketched bottleneck, the EMa Executor can be implemented in a fully distributed way, making use of MAS technologies (Wooldridge, 2002). The distributed approach runs all equal Executor instances (agents) in parallel on all robot cells at once, see Fig. 3.5. These agents communicate and negotiate/bid needs and resources what in fact implements the task allocation/resource assignment optimization process. All the processes can always be established within a maximum communication reach in terms of number of cells in a chain. The typical limits are - up to an end cell of a body branch or a malfunctioning routing cell. It is straightforward, that this setup allows all the interconnected (communicating) cells to create one logical unit and to stay always alive as a single robot. Thus the distributed approach overcomes the previous undesired behavior in the case of medial cell failure within a large robot body. Moreover, the MAS-based approaches allow treating specifically situations with communication inaccessibility within the system. In such cases, the inaccessible agents may be, up to a certain level, substitute by their models (stand-in-agents) to bridge temporary dropouts. To take into account varying conditions (i.e. to adjust task priorities important for the robot survival) MAS systems allow to modify negotiating strategy on the fly (Kulich *et al.*, 2007). Unfortunately, the latter MAS methods tend to be computationally intensive and therefore less suitable for use with the SYMBRION/REPLICATOR robots unless a light-weight implementation is developed.

3.1.12 Experiments

This EMa framework was tested on different platforms. The laser based localization and navigation Jockeys were tested in simulator Stage as a part of the Player/Stage system. The robot drives in the environment and on each crossing randomly chooses next corridor for traversing. An example run is depicted in Fig. 3.6. After the robot traverses predefined number of edges (20 in this case), the reasoning module starts its activity to close existing loops in the EMa structure.



(a) (b) (c)

Fig. 3.6 **(a)** Simulated environment; **(b)** gathered map; **(c)** Park pathway map.

The laser based localization and navigation Jockeys were also tested in an indoor environment. These experiments were performed with the G2Bot robotic platform equipped with two Sick LMS 200 laser range-finders in configuration where lasers provide whole 360° view. Alternatively, the robot was equipped with Hokuyo URG laser range-finder with 240° view.

The vision based localization and navigation Jockeys tests were reformed in real outdoor environment "Stromovka" park[1]. The park pathways were denoted by characters **A** to **W** (see Fig. 3.6) while the robot was manually guided by a human operator along them while EMa *SURFNav Learning Jockey* was recording detected arcs and vertices.

The size of the mapped area was approximately 400×300 *m*. While there is a need to guide the robot along each pathway in both directions, the total arc length was approximately 5 km. It took eight hours to guide the robot along all arcs and the resulting world representation created by the *SURFNav Learning Jockey* consisted of half a million recognized features. After that, the *SURFNav Navigating Jockey* was used to navigate the robot in the mapped environment.

The outdoor experiments were conducted with a Pioneer P3AT robotic platform with a TCM2 compass for heading determination. Other onboard equipment for the experiment was a Fire i-601c camera with 30fps in 1024x768 pixel resolution with

[1] Královská obora Stromovka, Prague 50°6'18.778"N, 14°25'33.395"E.

wide angle lens - focus length 3.5 mm. Obtained image data were processed in real time by an Intel Core 2 Duo 1.6GHz laptop.

All the tests were performed on more powerful hardware compared to the SYMBRION/REPLICATOR robot platform. The only purpose was to verify capabilities of the used approaches. Nevertheless, the achieved results have shown that the principal approaches used herein are well suitable in terms of the desired performance. The experiments have verified, that using the EMa approach, several "Jockey" agents can cooperatively explore the environment in order to build a distributed world model. The suggested methods remain subject of downscaling for the target HW platform and its computational power. Additionally, a special hardware for image-based world modeling (Svab *et al.*, 2009) can be considered.

3.1.13 Functional World Modelling

The SYMBRION/REPLICATOR robots perform tasks in an unknown and dynamic environment. The ability to build up a spatial world model is necessary for navigation, path planning and in general position-awareness. Such a spatial world model is however not sufficient to operate in such an environment. These tasks are embedded in a *general controller framework* (see Sect. 4.1). This controller framework can be a subsumption architecture or a more dynamic alternative, implementing for example a computational model for action selection (see Sect. 5.1.3). The controller architecture has to be operating on a world model. It has to interpret the world model. Granlund recognizes the importance of viewing a world model from a functional angle. In the article (Granlund, 2006) Granlund states:

> "Much of the lack of success in vision for complex problems can be traced to the early view that percepts should generate a description of the object or the scene in question. ... This description has typically been in geometric terms with a conceptual *similarity to CAD representations*. ... The major problem with this structure is that we primarily do not need a description of an object or a scene. What we need is an *interpretation* ..."

Objects and locations have spatial dimensions, however a SYMBRION/REPLICATOR robot needs to assign *functional roles* to objects and spatial entities. A hole in the wall can be interpreted as being big enough to crawl through. A block can be moved such that a power socket can be reached. This type of knowledge about the world is not accounted for by labeling objects in a 3D world model (with for example colour and other perceptual information). It constitutes a type of knowledge that is more easily implemented by *associative memory* models and non-spatial *sensorimotor learning* (see Sect. 3.2). This viewpoint can be summarized in this one-liner: *The world as a toolbox*.

A *labeled spatial world model* can be very useful, notwithstanding the fact that the labels are not a replacement for a functional context. The hippocampus is

often described as a collection of space cells elaborated with additional nonspatial information. Eigenbaum summarizes this (Eichenbaum, 2000):

> [S]patial firing patterns of these cells are strongly affected by the direction and speed of movement, by the targets of movement within the environment, and by demands of the behavioural test. Furthermore, hippocampal cell firing is also associated with many nonspatial events, including conditioned behavioural responses, olfactory cues and, in humans, categories of visual stimuli.

A functional world model extends a basically spatial representation with a functional dimension. This is contrary to adding a spatial dimension to a functional representation. The latter has been fruitfully applied in the form of *cognitive maps*. Even high-level cognitive phenomena like language can be considered as being spatial. The neurological substrate for spatial recognition, the hippocampus, is not just seen as a mere collection of space cells. It is just as well involved in episodic memory and declarative (relational) memory.[2]

The *acquirement* of a functional world model can be mediated by sensorimotor learning. Sensorimotor awareness (see Sect. 3.2) is not just about sensorimotor locomotion, sensorimotor aggregation nor about sensorimotor morphodynamics (this would include learning or modelling invariances between percepts and movement commands, aggregation commands and morphodynamic commands.) It is also about sensorimotor manipulation. This would enable the robot to move objects aside, crawl across complex obstacles, locomote through narrow passages, open doors, etc.

3.1.13.1 Problem Statement

A functional world model addresses the interface between a spatial, allocentric representation of the world and a repository of sensorimotor skills. In other words, the functional world model addresses the interactions between a cognitive map and procedural memory. This concerns sensorimotor skills that are not based on ordinary locomotion (this is accounted for in Sect. 3.1.3).

> How to adapt a spatial world model after achieving sensorimotor skills?

This research question addresses a transfer of meta-knowledge about learnt skills to the world model. If this type of meta-knowledge is available, we might be able to anticipate them:

> How to predict and anticipate changes in a spatial world model before performing a sensorimotor skill?

[2] Hippocampal involvement can be in the form of gating, filtering or selective learning. It does not mean that the representations are *located* in the hippocampus.

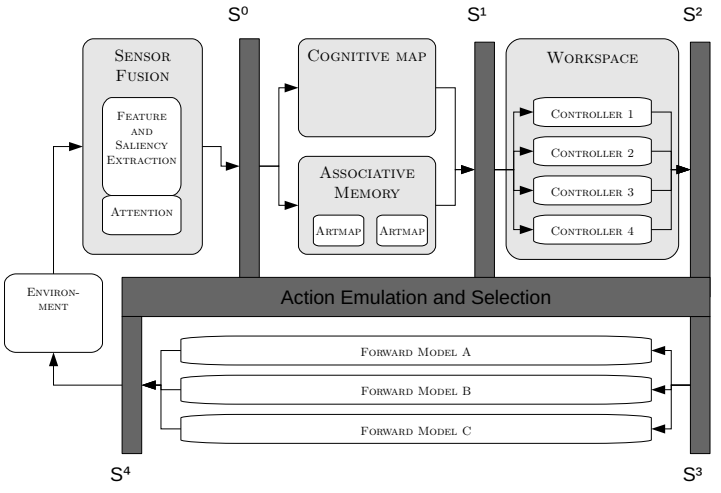


Fig. 3.7 The functional world model appends a spatial model (a cognitive map) with emulation capabilities. The emulator is visualized by a sort of H component with an additional upper leg. The cognitive map gets it input at S^0 which can originate from the sensor fusion modules or from the emulator. Between S^1 and S^2 different sensorimotor controllers can be emulated and the results tested with respect to the output of the cognitive map. In the emulation modus the forward models are enabled. In the runtime modus the output of the controllers results in actual actions in the real world.

A SYMBRION/REPLICATOR organism learns certain sensorimotor skills. What was impossible before this learning process becomes possible after. The organism learns to climb on a box, it learns to crawl through a hole in the wall. How is the cognitive map adapted when an organism obtains such a capability?

An organism should not blindly obtain sensorimotor skills. It learns them for a reason, for example to reach a place in the environment that was not reachable before. Performing a certain sensorimotor skill can change the spatial world model. If this change can be predicted, the organism can implement anticipation on this level. If an organism knows how the spatial world model changes when a box is moved, it can slide such a box under a power outlet.

3.1.13.2 Models

A collection of sensorimotor models have to be maintained in parallel. The use of paired forward and inverse models for motor control is part of sensorimotor awareness (see Sect. 3.2). This entails selective learning (Wolpert & Kawato, 1998), anticipation (Hoffmann, 2007) and (Pezzulo, 2008), planning (Toussaint, 2006). It is not only possible to emulate sequences of sensor-motor-sensor sequences. It is also possible to internally simulate higher level abstractions (this is done for higher-level sensor fusion, see Sect. 3.2). In the same way an internal simulation loop can use a forward model that predicts not raw sensor input, but new input for a spatial map (see Fig. 3.7).

The functional world model extends both the cognitive map (see Sect. 3.1.2) and low-level procedural memory (see Sect. 3.2). The benefit of such an integrated approach is, that it allows "reasoning" in spatial terms about the consequences of sensorimotor actions.

3.2 Emergent Cognitive Sensor Fusion

Anne C. van Rossum, Stephen P. McKibbin, Alfons H. Salden, Ted P. Schmidt

SYMBRION/REPLICATOR robots are modular robot organisms. They boldly go where no wheeled robot has gone before. The normally fixed architecture of traditional robots allows designers to make assumptions about their physical interactions and dynamics they experience. This may make the designer's job easier but it is at the cost of the adaptivity and flexibility of the final solution. Modular robots have the ability to change their architecture in a way that fixed uni-module robots cannot. They can purposefully adapt to their surroundings to enable better locomotion, sensing and recovery from damage. Where a wheeled robot might become stuck at an obstacle in its path, a modular robot can detect the problem and reconfigure its shape to allow it to crawl over the obstacle. The SYMBRION/REPLICATOR are different from other modular robots in a number of ways but not least due to the huge amount of sensors they carry. A model of emergent cognitive *sensor fusion* allows robots to make use of or dismiss this sensory data in a dynamic framework that works in synchrony with the robot organism, the environmental conditions and the current task being performed.

The robot organism is a modular organism. Contrary to many modular robots that have been built, the SYMBRION/REPLICATOR robots distinguish[3] themselves by the *plethora of sensors* they possess. Imagine a robot organism that consists of 10 robot modules. Each of those robot modules is equipped with many sensors (cameras, ultrasound sensors, microphones, infra-red sensors, laser scanner) and actuators (wheels, docking elements, LEDs). This amounts to large bandwidth requirements compared to robots that only use infra-red or bumper sensors[4]. Moreover, the docking possibility enables a robot to form an *organism* rather than a physically disconnected swarm. The sensor fusion model

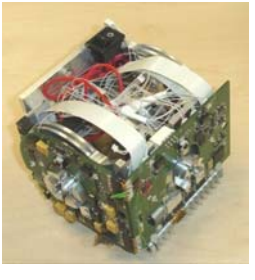


Fig. 3.8 One of the first prototypes of robot modules, see more in Sect. 2.1.

[3] The abundance of sensors and the powerful dual core BlackFin processor sets the robots apart from previous modular robots. There is no literature about such powerful modular robots. The literature describes mono-module cognitive systems, multi-module robots with simple, for example, infra-red sensors and swarm robots. There is no comparable framework that implements cognitive sensor fusion on robot organisms.

[4] One camera streaming images at a rate of 10 Hz, of a size of 256*256 pixels, coded by 3 8-bit RGB values amounts to a bandwidth of almost 16 Mbit/s.

needs to implement sensor *configuration*, sensor *data fusion*, sensory *learning* and sensory-motor *coordination* that fits the organism body form, as well as the task and the environment at hand.

The scientific challenge addressed in the sensor fusion model:

> Designing for emergent cognitive sensor fusion in a robotic organism that is subject to developmental, evolutionary and self-controlled metamorphosis?

Cognitive sensor fusion has to be performed on a modular robotic organism that changes its own body form — it exhibits metamorphosis. For that reason the cognitive sensor fusion architecture most likely has to change itself. Internal change, *internal metamorphosis*, is necessary to adapt to the outer changes of the robot body. The body changes can be governed by evolutionary means; caused by a development process or initiated by morphodynamic control. Moreover, as the body of the organism changes, so too does its exhibited behaviour. This can happen as the direct result of a corresponding change in control strategy or it can be as the result of the new dynamics the body shape has in interaction with the environment. The cognitive sensor fusion architecture must continually monitor and integrate the effects of morphology on the exhibited behaviour and the robot organism's perception.

The proposed architecture consists out of five layers (Fig. 3.9). The lower three layers address the topic of sensory-motor and sensor fusion:

- The sensory-motor layer is described in Sect. 3.2.3.1. It is built on top of the *distributed forward models* by Tani, see (Tani, 2005);
- The sensor fusion layer is described in Sect. 3.2.3.2. It uses the *saliency-based Itti, Koch and Niebur attention model*, see (Itti *et al.*, 1998), an *adaptive resonance theory model as associative memory* (Grossberg, 1987) and *global workspace theory for anticipation* (Baars, 1988);
- The eco-devo layer is described in Sect. 3.2.3.3. It uses a *gene regulatory network for (genetic) indirect encoding* of the sensor fusion layers (Bongard & Paul, 2001), (Quick *et al.*, 2003).

This section will describe sensor fusion that survives metamorphic changes on an artificial organism. It will *not* describe metamorphic *control* itself! It makes a considerable effort to provide a framework that can be used by controllers, without touching actuators[5]. This section addresses a modular robot organism, in which modules have to cooperate in fusing incoming sensor data. It does *not* address a robot *swarm*.

The rest of the section is organised as follows. Sensory-motor awareness and sensor fusion *scenarios* illustrate what the sensor fusion model needs to be able to implement (Sect. 3.2.1). The background of *embodied* cognition and *emergent* sensor fusion is described in Sect. 3.2.2. And the actual *integrated model* for cognitive

[5] If the sensor fusion layer would drive the wheels or turn on the LEDs, conflicts with higher layers have to be accounted for.

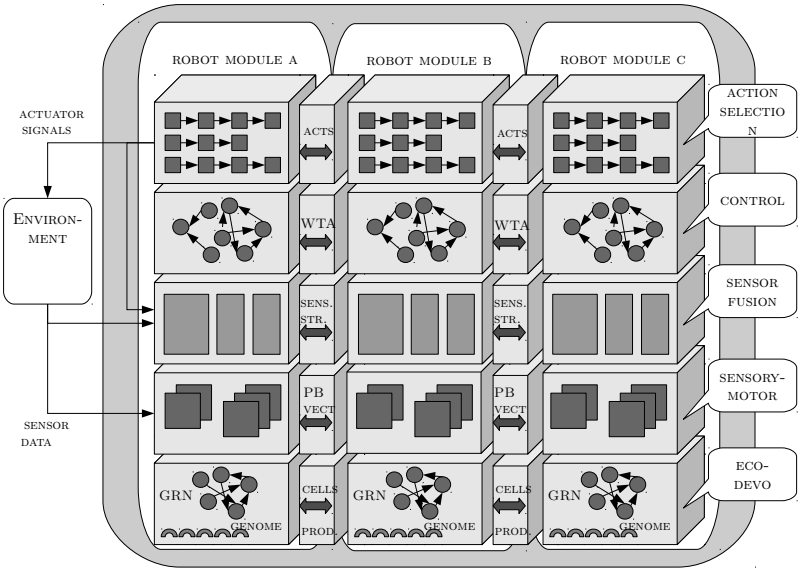


Fig. 3.9 The proposed sensor fusion architecture in the context of other functionality on the organism. At the bottom an eco-devo engine — using a gene regulatory network (GRN) — allows for a grown distribution of modules across the organism. The sensory-motor layer adds sensory-motor categorisation and prediction. The sensor fusion layer adds multi-modal associations, as well as attention. The controller layer contains controllers and a winner-take-all (WTA) mechanism. The action selection layer translates control into a series of actions that has to be undertaken or internally simulated.

emergent sensor fusion on a robot organism, that is being developed, is presented in Sect. 3.2.3.

3.2.1 Scenarios

As discussed in Sects. 1.1 and 2.2.7, two "Grand Challenges" for self-reconfigurable artificial organisms robotics are defined. One of them is related to a long-term independency and self-sufficiency in an unknown and changing environment, where essential tasks for energy harvesting include recognition and localisation of power sources, collective locomotion, collective reaching and energy sharing. This "Grand Challenge" is the *background setting* for the following sensory-motor categorisation and sensor fusion scenarios. The scenarios described below are arranged in increasing complexity starting with relatively simple yet dynamic tasks focusing on physical interactions to more elaborate tasks that require morphing and collaborative sensing.

Pushing Walls

The envisioned sensory-motor skills on modular robotics range from *multi-robot organism locomotion* and *purposeful manipulation of objects* in the environment, to

cognitive capabilities such as *learning sensory-motor* patterns and *anticipation of self-movements*. Those scenarios test specific design principles for sensory-motor awareness. Pfeifer et al.'s robotic agent design principles has been used as a guideline (which will be explained later in Table 3.1).

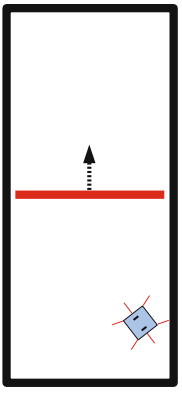


Fig. 3.10 A single robot in a walled arena. The pushable wall is only highlighted red for clarity but it appears identical to other walls. The robot must interact with the arena to establish the physical dynamics of its environment.

When interacting with its environment, a robot, or any embodied agent, will experience various sensory stimuli. This stimulation maybe internal (temperature, current in motors, sound), proprioceptive (joint torques, joint encoders, balance) or it may be external (external sound, vision, pressure, tactile). These stimuli may occur as the result of the agent acting in the environment causing self-stimulation (active) or being acted upon by something in the environment (passive) or a combination of both. These 5 factors of internal, proprioceptive, external, active and passive sensory experiences intertwine to produce a perception experience for the agent. Moreover, the way that the agent detects stimulus in its environment is also very context dependent and can rely on the "intentions" of the agent at the time. It has been shown that descending motor commands can have a large influence on sensory information that is sent from the periphery to the central nervous system in humans/animals (Cullen, 2004).

When a robot is moving in its environment it is generating internal, proprioceptive and external sensory sensations and often it is useful for the robot to be able to integrate these sensations to determine something about the environment or the situation it is not experiencing. For example, if a robot is trying to explore its environment it may have to *establish the dynamics of the objects in it* (of which it itself is one).

Coordinated Locomotion

An extension of this scenario is in coordinated locomotion. A robot should be able to coordinate its actions with another conjoined robot without necessarily sharing its intentions, or motor commands, and sensory information (internal and external). This way, coordinated movement can emerge from the interaction of two *logically separate, but physically connected* agents. Through this agent, environment, control system interaction, computationally difficult tasks can be achieved. For a number of conjoined robots to move in a coordinated manner they must be able to cooperate in a useful way and also possibly combine such coordinated movement with other behaviours like obstacle avoidance (repulsion) or wall pushing (attraction).

This task can be achieved by communicating between the robots what each robot is doing, what each robot is sensing and what each robot is trying to do. An

alternative solution to this problem would be to solve in a distributed manner. This approach negates the need for extensive communication and thus the processing of the data that is received from each robot. By relying on the fusion of information that is available to each robot through the wheel encoders, motor commands and infra-red sensors, robots can interact with each other and, through this interaction, they produce coherent behaviours.

Articulated Chain

The coordinated movement scenario that is outlined above is based on a number of robots that are physically coupled at their sides, or "joined at the hip". Another extension to this scenario is based on the coordinated movement of a more articulated robot organism. This organism again consists of a number of conjoined robots but this time, instead of each robot being connected at its sides, every other robot in the line is connected at its front and back and is rotated by 90 degrees to make a hinge joint perpendicular to the floor. These hinge joint robots do not contribute to the forward motion of the chain but simply facilitate its articulation. This articulated chain is able to coordinate its movement in order to locomote quickly and effectively on a flat (or quite flat) surface. This task will allow the organism to *surround an object or cordon off a dangerous area*. Sensory-motor fusion must take place in order for the robot organism to navigate towards the target or target area by fusing information from docking sensors, wheel encoders, joint encoders, infra-red sensors, camera as well as motor control commands.

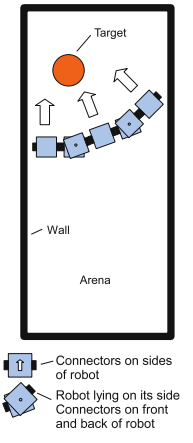


Fig. 3.11 Several robots joined together to make a chain. The linked robot chain is made articulated by the joining robots turned on their sides that can bend in the middle.

Argus Panoptes scenario

The Argus Panoptes scenario refers to the Greek god Argos who had so many eyes that there were always some of them open. An organism contains sensors on each module, so it is important to have sensors turned off to reduce the bandwidth of processing all that data.

Fig. 3.12 The flatworm-like or snake-like organism in the simulator. After cutting it in two parts, both parts should still be able to pursue its goal of following a moving target.

In the Argus Panoptes scenario a robot organism can be approached from different directions by a moving object. The robot has to *detect the incoming target as quickly* as possible. The organism works in its entirety as a rocket shield where each module does have its role in detecting incoming missiles as quick as possible. This scenario does not pose locomotion demands or locomotion agility demands. Experiments will probe

the ability of the robot to *disregard* sensory information. This scenario necessarily includes attention mechanisms.

Planarian scenario
The Planarian is a flatworm that when divided into two parts, regenerates both parts, so that the result contains two full-grown Planaria. Moreover, there is evidence that an unconditional response that is learnt before the fission moment, is *still remembered after the division* (Mueller & Levin, 2002). The Planarian scenario contains a robot snake following a moving target (see Fig. 3.12). Subsequently a human operator issues a split event which is performed by the middleware. The resulting two snakes should still be able to follow this moving target on their own. Their sensory perception should not be so severely hampered that they can't perform this task any more. Performance in this scenario is related to the gradual degradation of the robot's perceptual capabilities when the hardware becomes limited.

Musca Domestica scenario
The Musca Domestica, a fly, remembers olfactory information across metamorphosis (Ray, 1999). The metamorphosis process is autonomous and self-initiated.

The Musca Domestica scenario assumes a robot that actually can change its body form[6]. Both forms correspond to an entirely different way of locomotion. The robot morphs from spider to snake, and the other way around (see Fig. 3.13). The scenario contains two phases. In the first phase a robot learns to follow a certain object of specific visual-acoustic properties by positive reinforcement. In the second phase, the robot is able to follow this object, while reversible (back-and-forth) metamorphosis occurs during this task. Observe that this scenario demands *metamorphosis of the internal sensor fusion architecture*.

Fig. 3.13 The spider-like organism in the simulator. It should be able to retain learnt patterns even after morphing to a snake-like organism and morphing back.

3.2.2 Towards Embodied and Emergent Cognition

Artificial organisms conceive of their environment in a situated, and even an embodied way. However, what does this term embodiment mean?

Types of embodiment
Ziemke discusses the types of body that might be required for *embodied cognition* and whether a body is actually needed at all (Ziemke, 2002). He outlines six types of embodiment as follows:

[6] Observe that body metamorphosis is (again) a given for sensor fusion. Body changes are covered by the control layer. Cognitive sensor fusion is concerned with surviving such external changes of the body.

1. *Structural Coupling* - at any time, the states of two embodied systems can be, and are, affected by perturbation from the other;
2. *Historical Embodiment* - a systems embodiment is the result of a historical process that results in its current embodiment;
3. *Physical Embodiment* - as opposed to software agents, embodied agents require a physical body that can operate in the real world;
4. *"Organismoid" Embodiment* - the physical body of an agent should have organism-like qualities such as body shape, sensory-motor capacities, etc;
5. *Organismic Embodiment* - only biological bodies perform cognition. There are fundamental differences between natural and man-made bodies. Natural bodies grow into collections of specialised organs, machines are built from pre-designed specialised parts;
6. *Social Embodiment* - social interactions have an effect and are affected by the embodiment of others. This relates to the production and interpretation of body language.

An artificial organism is embodied in a structural, physical and organismoid sense. The sensory-motor and sensor fusion layers *learn* respectively sensory-motor and sensory categories (see Sect. 3.2.3). This introduces *historical* embodiment. A bio-inspired process *grows* a set of modules in the eco-devo layer (see Sect. 3.2.3.3). A borderline case of *organismic* embodiment. In the sensor fusion model the organism mode is emphasised above the *swarm* mode however *social* embodiment will still be present when robot organisms interact with each other for purposes of docking or even for imitative learning.

Robotic design principles

A number of design principles have been laid out by Pfeifer et al. for building intelligent robotic systems (Pfeifer *et al.*, 2005a).

The list of design principles are divided into two types:

1. Design procedure principles: these are a general methodology one should apply in approaching the design of the *system as a whole*;
2. Agent design principles: these principles are guidelines for designing the *intelligent agents* themselves.

The design procedure principles apply to creating a SYMBRION/REPLICATOR robot as an intelligent adaptive, embodied system. P-Principles 2, 3 and 4 describe the self-organising aspect of creating an organism. The configuration of the robot, at any given point in time, is the result of many decoupled processes occurring dispersed across the system in accordance to the affordances or constraints of the environment. These constraints tend to have the effect of imposing coordination on these processes in order to improve the overall system performance. The system will evolve over long time scales, while it will learn and physically change shape over short time scales. The model largely relies on the A-Principles 3 and 4. Loosely coupled processes (Sect. 3.2.3.2) and sensory-motor coordination (Sect. 3.2.3.1).

Table 3.1 Robotic agent design principles by Pfeifer et al. The P-principles 1 till 5 describe a general system design methodology. The A-principles 1-8 describe specific aspects of agent design that should be attended to.

Label	Name	Description
		Design procedure principles
P-Princ 1	Synthetic methodology	Understanding by building
P-Princ 2	Emergence	Systems should be designed for emergence (for increased adaptivity)
P-Princ 3	Diversity-compliance	Tradeoff between exploiting the givens and generating diversity solved in interesting ways
P-Princ 4	Time perspectives	Three perspectives required: "here and now", ontogenetic, phylogenetic
P-Princ 5	Frame of reference	Three aspects must be distinguished: perspective, behaviour versus mechanisms, complexity
		Agent design principles
A-Princ 1	Three constituents	Ecological niche (environment), tasks, and agent must always be taken into account
A-Princ 2	Complete agent	Embodied, autonomous, self-sufficient, situated agents are of interest
A-Princ 3	Parallel, loosely coupled processes	Parallel, asynchronous, partly autonomous processes, largely coupled through interaction with the environment
A-Princ 4	Sensory-motor coordination	Behaviour sensory-motor coordinated with respect to target; self-generated sensory stimulus
A-Princ 5	Cheap design	Exploitation of niche and interaction; parsimony
A-Princ 6	Redundancy	Partial overlap of functionality based on different physical processes
A-Princ 7	Ecological niche	Balance in complexity of sensory, motor, and neural systems; task distribution between morphology, materials, and control
A-Princ 8	Value	Driving forces; developmental mechanisms; self-organisation

Emergent sensor fusion

An organism consisting of 10 modules contains a lot of sensors. The most recent module prototype (see Fig. 3.8) contains 4 cameras, a laser, microphone and infrared sensors. A robot of 10 modules would contain, a stunning, *40 cameras*! There is a need to restrict this sensory stream in a body-aware and situation-aware manner. This is accounted for in two ways in the sensor fusion model. *Cognitive capabilities* like attention, multi-modal association and anticipation are incorporated in the sensor fusion layer. This allows for turning off entire sensors, but it also allows for disabling sensor fusion filters on a finer granularity. The eco-devo layer implements a *growth process* of the modules within the sensor fusion layer. This allows for adaptation of the sensor fusion modules to the growing and morphing robot body.

Table 3.2 The differences between cognitivist and emergent cognition as summarised by Vernon et al. The approach undertaken in this work, pursues the emergent paradigm. Sensor fusion as considered as a concurrent self-organising networked process, with which comes a lot of other characteristics that are exemplary for the emergent approach. *Republished with kind courtesy of the primary author David Vernon.*

Cognitivist *versus* Emergent Paradigms of Cognition		
Characteristic	**Cognitivist**	**Emergent**
Computational Operation	Syntactic manipulation of symbols	Concurrent self-organisation of a network
Representational Framework	Patterns of symbol tokens	Global system states
Semantic Grounding	Percept-symbol association	Skill construction
Temporal Constraints	Not entrained	Synchronous real-time entrainment
Inter-agent epistemology	Agent-independent	Agent-dependent
Embodiment	Not implied	Cognition implies embodiment
Perception	Abstract symbolic representations	Response to perturbation
Action	Causal consequence of symbol manipulation	Perturbation of the environment by the system
Anticipation	Procedural or probabilistic reasoning typically using *a priori* models	Self-effected traverse of perception-action state space
Adaptation	Learn new knowledge	Develop new dynamics
Motivation	Resolve impasse	Increase space of interaction
Relevance of Autonomy	Not necessarily implied	Cognition implies autonomy

The literature (Vernon *et al.*, 2007) makes a distinction between cognitivist and emergent cognition. In the sensor fusion model, the *emergent paradigm* will be followed. As can be seen from Table 3.2 by Vernon et al. this involves self-organising networks, rather than syntactic manipulation of symbols. In the context of cognitive sensor fusion, this means that fusion takes place by an *interacting network of sensor fusion agents*. The world is not directly interpreted as a set of symbols: doors, birds, left, right, red, blue, robot, human. Instead of that, agents deliver functionalities on the level of sub-symbolic feature maps.[7] The feature maps can be combined on a sub-symbolic level and directly drive the motors. Perception is represented in an emergent paradigm by raw perceptual input rather than perceptual symbols. The sensor input does not need to be translated into a symbolic representation in this case. However, higher level controllers might pose a symbolic representation as a

[7] An example of what is meant by a sub-symbolic feature map, is the saliency-based map described in 3.2.3.2.

prerequisite for their operation. How such symbolic or category labels can be learnt in an emergent paradigm is described in the end of Sect. 3.2.3.1 about the sensory-motor layer.

3.2.3 Sensor Fusion Model

The sensor fusion model concerns itself with the lower three layers in Fig. 3.14 (see also the introduction):

- The sensory-motor layer is described in Sect. 3.2.3.1. It is built on top of the *distributed forward models* by Tani, see (Tani, 2005);
- The sensor fusion layer is described in Sect. 3.2.3.2. It uses the *saliency-based Itti, Koch and Niebur attention model*, see (Itti *et al.*, 1998), an *adaptive resonance theory model as associative memory* (Grossberg, 1987) and *global workspace theory for anticipation* (Baars, 1988);
- The eco-devo layer is described in Sect. 3.2.3.3. It uses a *gene regulatory network for (genetic) indirect encoding* of the sensor fusion layers (Bongard & Paul, 2001), (Quick *et al.*, 2003).

3.2.3.1 Sensory-Motor Layer

The robot needs to be able to categorise what it is doing as well as what it is perceiving. By integrating the intent of carrying out an action (a motor command) with the resulting sensory stimulus (proprioception or external sensing), context can be

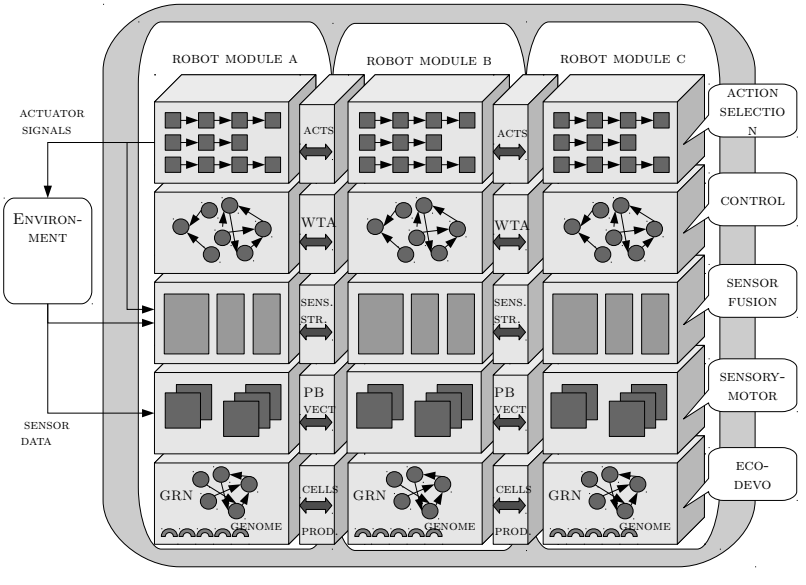


Fig. 3.14 The proposed sensor fusion architecture. *Repetition of Fig. 3.9.*

developed for the robot organism. For example, in the *wall pushing* scenario, the robot should be able to establish that a wall is pushable if its wheels are turning, it senses an obstacle and the command to go forward was issued (disregarding wheel slip). Without the information that the robot had issued the *go forward* command, the context could not be established. It would be unclear if the robot was pushing the wall or the wall was pulling the robot. It is necessary that the robot's perceptual system integrates the bodily aspect of sensing not just the environmental or external aspect. The sensory-motor layer is concerned with integrating sensory-stimulus that is generated by the robot's own actions along with the commands that caused them and also the external stimulus that arose from the environment.

Natural organisms are able to integrate sensory-motor information in a context dependent manner. A version of what is known as the *efference copy* model (von Holst & Mittelstaedt, 1950) suggests that when motor commands (efferent signals) are issued, a *copy of the command signal* is sent, or held, for *internal processing* (see Fig. 3.16). The signal that returns, the afferent signal, should correspond with the command that was sent unless an external event or force has caused the command to not be executed. Subtracting the afferent signal from the efferent copy the result should be zero (or below some threshold) otherwise an external event has occurred and the resulting difference signal (reafference) should be processed. One benefit of using this method is that *self-generated and externally generated movements can be distinguished*. Viewed another way, the organism has an inherent model of how its body works and how it affects its sensory experience. With the "efference copy" model, the organism can also determine how the environment affects its sensory experience.

Forward model. It has been suggested that vertebrates and invertebrates may use a more sophisticated method known as a *forward model* to predict the possible outcome of certain inputs given a particular context (Webb, 2004). In the case of forward models, the organism uses knowledge of its sensory-motor system in order to predict the resulting sensation it should experience when a motor command is issued in a given context. This model holds information on the sensory-motor flow and produces a mapping that is capable of outputting expected sensations. Using this expected sensation, that is calculated by the forward model for a given motor command, an organism can determine if an action command was carried out intentionally or if any sensation was due to outside factors.

Forward-inverse pairs. Wolpert and Kawato have proposed using a forward model that is able to learn the dynamics of sensory-motor system by minimising the error between the expected input and the actual sensed input at the next step (Wolpert & Kawato, 1998). As discussed previously, this model allows the system to have a mapping between its sensory and motor apparatus and it can establish causality for its sensations. The authors propose using a different forward model for each sensory-motor behaviour, in essence representing sensory-motor primitives. Each *forward model has a paired inverse model* (or controller) that is able to produce the motor commands that would be required to generate the desired sensation for the given context. By matching each inverse model to a forward model the

system is able to determine which model to use at any point in time from the prediction error generated by the forward model.

Distributed model. The model described by Wolpert and Kawato uses what is known as a "local representation scheme" for encoding each sensory-motor primitive (Wolpert & Kawato, 1998). In this scenario, when a new primitive is to be learnt, a new forward-inverse pair must be created to include it in the system. Tani suggests that this is untenable and that the system would require large numbers of models in order to produce the necessary sensory-motor behaviours (Tani, 2005). Instead, Tani suggests to use a *distributed model* where all the sensory-motor mappings are mapped in a single neural network, a so-called Recurrent Neural Network with Parametric Bias (RNN-PB). When additional mappings are to be learnt, they are distributed across the network according to their relationship with already stored mappings. In Tani's model, the recognition of sensory-motor patterns is achieved by performing an *inverse regression operation* in order to produce the correct Parametric Bias (PB) values that correspond with the current sensory-motor pattern. This is not a biologically plausible operation whereby a network steps back through time in order to settle to the correctly recognised PB value.

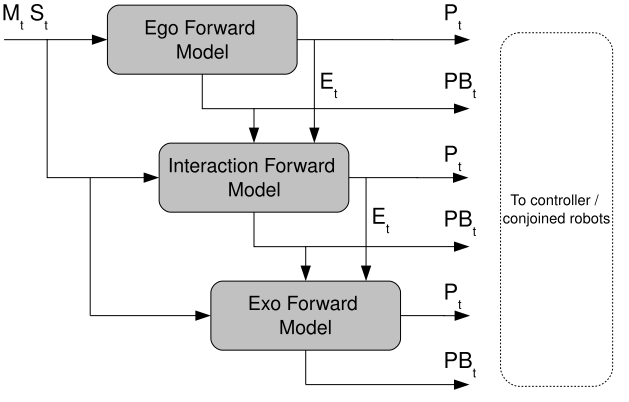


Fig. 3.15 M_tS_t = Sensor-motor flow, P_t = Predicted sensory-motor flow, E_t = Prediction error, PB_t = Parametric bias vector associated with sensory-motor flow. 1.) The Ego Forward Model is based on the *self-generated stimuli* of the body. This is trained using a single robot module as this unit is indivisible and remains constant. The Ego Forward Model predicts the bodily sensations caused by bodily actions. 2.) The Interaction Forward Model predicts the sensory consequences of interactions with the environment that includes *exogenous (external) sensing*. External senses are also affected by the actions of the robot, for example, the variation of light stimulus as the robot bends in the middle pointing its sensors upwards. The Interactive model receives the error signal from the Ego model, which describes the part of the stimulus that is not body based, and the PB value, which describes the model's "best guess" as to what is being perceived. 3.) The Exo Forward model is concerned with predicting sensations that are *passively experienced*, for instance, the resulting stimulus caused by effects originating outside the body.

The sensory-motor layer will use features of both the distributed model and the local representation scheme. It is proposed that the distributed representation scheme, used by Tani et al, will be useful to learn *relational structures in the sensory-motor flow* of the robot organism. Also, the ability to associate these distributed representations with a point in the self-organised PB space makes it possible to chunk many sequences together at higher levels of granularity. The local representation scheme is useful to establish individual models for certain aspects of the learning process *without interfering with already learnt models*. This ability is to be utilised in the system by implementing different aspects of sensory-motor learning across different models of *increasing abstraction*. This is shown in Fig. 3.15. There are three different abstraction levels:

1. The Ego Forward model learns the relational structures between an individual robot's actions and the generated sensory feedback. Such feedback will be *purely internal* and based on modalities such as wheel encoders, joint encoders, gyroscope and possibly force sensors. By implementing a process of "body babbling" the relational structures between control commands and sensory feedback can be learnt and predicted by the model. Any deviation from this model implies that there is something else in the environment causing new, unknown, sensations;
2. Another model, the Interaction Forward Model, learns to associate the sensory-motor flow with the deviations from the Ego model. This association allows the sensory-motor system to predict and represent the *active interactions* that it has with both the environment and with objects in it. For example, the movement of the robot will impinge on its wheel encoders, but also on its cameras and distance sensors. A robot that is physically linked to another robot will receive information of this link by the docking sensor but also it will experience forces at its joints caused the additional mass of the other robot. Sensory-motor categories can be learnt, such as *isPushable*, *isRollable*, *isTraversible* (bridge), *isEnterable* (door);
3. The final model is the Exo Forward Model that is used to establish and predict the *passively experienced dynamics of the environment*, such as a rolling ball, a falling object or a swinging door. Moreover, the prediction of the actions of another robot may be useful in collaborative tasks such as docking. The ability to predict the actions of another robot may also prove to be critical if the robot is to perform imitative learning.

Symbolic sensory-motor categorisation. The sensory-motor layer does not necessarily have to communicate with the controllers with PB vectors. Tani et al. introduce the possibility of learning sensory-motor categories (Tani *et al.*, 2004). This is shown in Fig. 3.16. In this setup, there is a separate set of inverse and forward models that learn symbolic category names in parallel with PB vectors. A higher level controller can be using those category names instead of PB vectors that indicate, for example, the isPushable, isRollable and isTraversible primitives mentioned above. Moreover, in this scheme, a higher level controller can interface with the learnt categories by instantiating the mirror-system capabilities of the forward-inverse model. For example, the controller can use the symbolic representation of

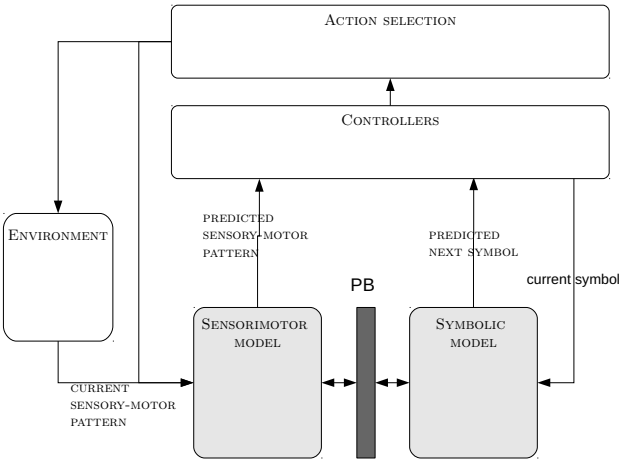


Fig. 3.16 The sensory-motor layer. The model communicates internally with parametric bias (PB) vectors (Tani *et al.*, 2004). The PB vectors are abstractions of sensory-motor input. In this figure, a symbolic/linguistic module allows to send not only PB vectors as abstractions, but also category names. Those category names (symbols) can be used by controllers to label sensory-motor skills.

the learnt sensory-motor sequence that allowed the model to establish *isPushable* to initiate that particular action sequence. In this case the controller is using the learnt inverse model to generate a desired sensory-motor sequence from its associated representation. This facility is not the focus of the sensory-motor categorisation as it incorporates actuation into the process however it might be useful to investigate the role of actuation in perception or *active sensing*.

The symbolic categories highlighted above are subjective in the sense that they involve an interaction between the robot and its environment. It is foreseen that it will also be useful for the robot to identify certain aspects in its environment in a more objective way. The robot is not a "blank slate", it will need to possess some innate abilities to categorise sensory stimulus. For example, one aspect of the environment that is predictable is the fact the individual robot modules will share their environment with other modules. On this basis we can assume that the robot should be able to identify their conspecifics as it is almost guaranteed that they will meet and interact at some stage. This is an example of an innate category that the robot should recognise. Another innate categorisation that is useful for the robots is that of a power socket. The "Grand Challenge" explains that the robot system should "survive" without human assistance and this will inevitably require the ability to identify energy sources (or at least potential energy sources). This categorisation ability still requires the integration of a subjective element in its implementation because the current actions of the robot will still affect the sensory stimulus associated with objects such as robots or power sockets. Other such categories will be the focus of investigation as the project progresses.

3.2.3.2 Sensor-Fusion Layer

The sensor fusion layer is divided into three large blocks: Feature and saliency extraction; attention mechanisms and cross-modal associations (see Fig. 3.17).

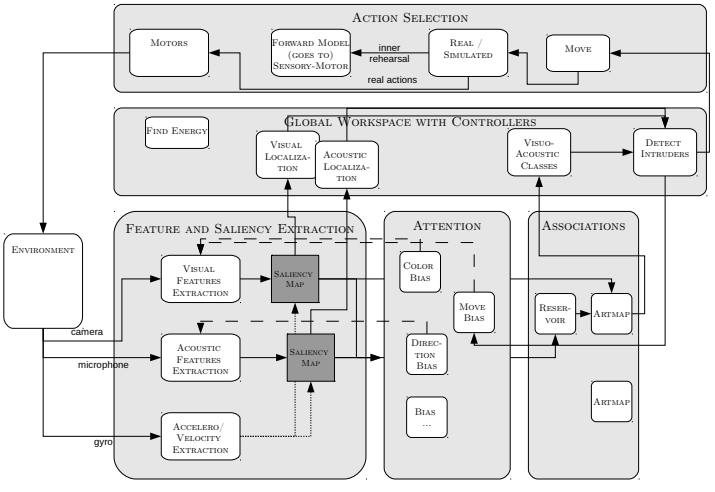


Fig. 3.17 The sensor fusion layer. The sensor fusion layer is divided into three large blocks. The first block implements feature and saliency extraction. The second block contains attention mechanisms. The third block enables cross-modal associations.

There are four cognitive capabilities in the sensor fusion model:

- *Attention*: the ability to focus on certain visual, acoustic, tactile objects, and to switch attention from one entity to another;
- *Association*: the ability to associate information from different modalities to be able to recognise, for instance, visual-acoustic entities;
- *Anticipation*: the ability for inner rehearsal, sensor fusion categories lead to simulated actions, which lead to sensor fusion categories, etc;
- *Awareness*: the inclusion of systemic sensors;

Cognitive Models. There are two famous cognitive architectures that are worth mentioning. The *Kismet robot* is developed at MIT (Breazeal, 2003). In Kismet, the perceptual system processes visual and auditory information. It performs feature extraction, focusing on, amongst others, the emotional "vibe" of speech. Acoustic processing is specialised to detect emotional undertones in speech. The architecture does not address the fusion process between vision and audition. It does however address an *attention mechanism* that is able to bias the system's interest for faces versus objects (Breazeal & Scassellati, 1999). This attention model is derived from the Itti, Koch and Niebur (IKN) model, see below. The Kismet architecture implements attention and awareness, but does not address association or anticipation explicitly.

Another famous cognitive architecture is the *iCub cognitive architecture*[8]. The iCub architecture contains a network of competing and cooperating distributed multi-functional subsystems, resembling a multi-agent system. Subsequently, important characteristics are implemented like adaptation, self-modification, anticipation and embodiment (in the form of crawling, sitting, reaching, grasping, imitation and social interaction). The cognitive architecture in the iCub robot is very explicitly based on sensory-motor skills (see Sect. 3.2.3.1). Those skills can be modulated or inhibited by a less reactive system. This modulation system is formed by auto-associative memory that builds percepts out of different modalities, a motivation system and an action selection module. Besides that there is a second-order loop that simulates the outer world and rehearses sequences of sensory-motor events. In this way the overall system is able to anticipate the consequences of its own actions. The iCub architecture implements association, anticipation and awareness, but does not address attention explicitly.

The *sensor fusion* model addresses all cognitive capabilities, from attention, association, anticipation to awareness.

Attention. The sensor fusion model implements the saliency-based sensor fusion model as described by Itti, Koch and Niebur (the IKN model). The IKN model has been studied and applied multiple times (Itti *et al.*, 1998), (Itti & Koch, 1999), (Itti & Koch, 2001a), (Itti & Koch, 2001b). The model combines different visual sub-modalities, like colours, intensity and orientations into an overall saliency map. Subsequently, a winner-take-all network defines what spatial position on this map will be considered as the next focus of attention. The weights of the different sub-modalities can be adjusted in the process. This gives the opportunity to steer those weights by a top-down attention mechanism. It can bias the system for colour or orientation and it enables the robot to prioritise moving objects (like other robots) above static objects (like energy sockets) in an office environment.

Association. Sensor data fusion concerns the creation of a fused representation of several sensors from different modalities. A fused representation does not necessarily need to be local e.g. an associative memory maintains a distributed representation. Integration of sensory information is achieved by dynamic binding[9] rather than fusion into a more abstract representation. Such an associative model is Adaptive Resonance Theory (ART) by Grossberg. This theory dates back to 1976. There are versions that perform supervised, as well as ones that perform unsupervised, learning (Grossberg, 1987), (Carpenter & Grossberg, 1994). Unsupervised means that it only extracts statistics from the offered input patterns and stores those in a (preferably) compact representation. It has benefits to do this, because it is for example possible to offer "input patterns" that are incomplete and the system will be able to "fill in the gaps". Basically, Grossberg's approach was an attempt to fight the

[8] The iCub cognitive architecture is developed in a European FP6 project called RobotCub.

[9] Binding is meant in a neuroscientific sense rather than a computer science sense. It concerns the *binding problem*: "When I see a blue square and a yellow circle, what neural mechanisms ensure that the sensing of blue is coupled to that of a square shape and that of yellow is coupled to that of a circle?".

problem of *"catastrophic forgetting"* or the *"stability-plasticity dilemma"*. After learning pattern A, as well as pattern B, the former pattern A should not be forgotten by the system.

Anticipation. A *global workspace* for controllers allows for anticipation, see (Baars, 1988) about global workspace theory (GWT). Implementations of GWT (Shanahan, 2006), (Franklin & Ferkin, 2006), allow an organism to emulate motory-sensor patterns in advance. The cognitive architecture contains data processing modules as well as associative memory. Besides these, it needs forward models to implement this type of sensory forecasting. The forward models predict the next sensory input on a certain actuator output. Forward models can be learnt as explained in Sect. 3.2.3.1 about the sensory-motor layer. However, forward models do not necessarily have to map *raw motor commands to raw sensory* data. For instance, shifts of the camera when the robot moves. No, on a cognitive level it operates on a more abstract level. A *multi-modal percept is predicted*, rather than raw visual data. It is an amodal emulator. Chains of tasks and higher-level percepts are emulated beforehand, disregarded and finally accepted. This implements anticipation.

Awareness. The inclusion of systemic sensors, makes a multi-modal percept also aware of the *internal state* of the robot. Awareness is implemented in the sensory-motor layer (Tani, 2005). Hence, the cognitive sensor fusion architecture contains attention, association, anticipation, as well as environmental and self-awareness mechanisms.

3.2.3.3 Eco-Devo Layer

There are evolutionary and morphodynamic controllers that control the shape of the organism due to changes in the environment or the task at hand. The multitude of different shapes leads to a combinatorial explosion of *ways in which data can be processed* within the organism. A camera that is continuously seeing the organism's own body, might be turned off, or it might focus on colour rather than motion. A so-called eco-devo layer allows the sensor fusion model to code for cognitive sensor fusion components and their interactions, see Fig. 3.18.

There are several encoding schemes within genetic algorithms. They differ in their compressibility[10], use of self-organisation and dynamics of the resulting topology. The term *sensor fusion topology*, or the shorthand topology, has to be understood as a directed graph in which nodes denote sensor fusion components (like a sensor, a saliency map, a bias component or an adaptive resonance theory module) and edges denote weighted connections between those components.

The direct encoding scheme is for example used in *evolution strategies* (Salomon, 1996). Evolution strategies implement coordinated mutations. An example of indirect encoding scheme is HyperNEAT, Hypercube-based NeuroEvolution of Augmented Topologies, (D'Ambrosio & Stanley, 2007),

[10] A computable object is called compressible in computer science, if there is an algorithm that computes it, but has fewer bits. This is related to Kolgomorov complexity.

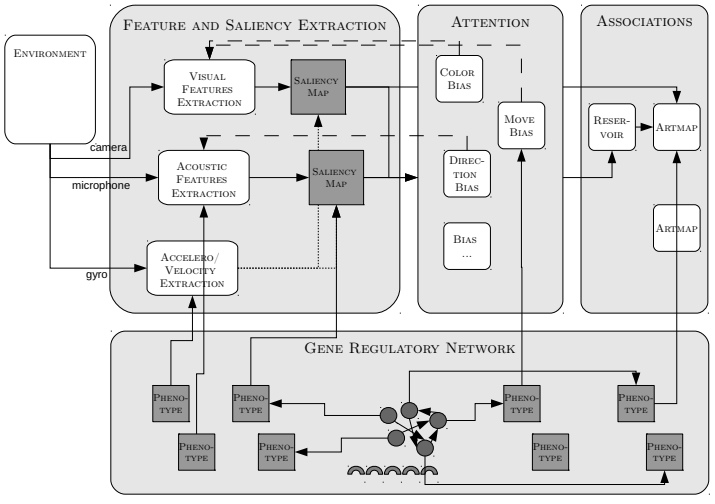


Fig. 3.18 The eco-devo layer. The picture shows which elements of the sensor fusion layer can be genetically represented. The picture has to be visualised on each robot module in the organism. At the far right, individual sensors can be turned off, by disabling the feature extraction module. The saliency map option, and, at the left, the biasing filters can genetically code preferences for movement versus colour, or vision versus sound. At the far left, associative networks can be organised in a genetically defined hierarchy.

(Gauci & Stanley, 2008). HyperNEAT allows for compression in the form of *gene reuse* (Inden, 2007), (Roggen & Federici, 2004), (Kumar & Bentley, 2000). One of the indirect encoding schemes that uses self-organisation are *gene regulatory networks* (GRN). Such a self-organised encoding scheme provides possibilities to *interact with the environment* (Pfeifer *et al.*, 2005a), (Inden, 2007). The eco-devo approach takes this one step further, so, that this interaction between the genes and the environment *never ceases* (Quick *et al.*, 2003). This means that game-of-life-like morphing behaviour can be implemented.

The sensor fusion model uses the eco-devo approach. The continuous morphing of the body of the artificial organism is analogue to the process of *metamorphosis* in nature. The corresponding morphing of the *internal* architecture of the robot is called *mesomorphosis*. Mesomorphosis addresses the adaptation of a caterpillar brain to a butterfly life. Artificial metamorphosis or mesomorphosis does not stop when a certain body morphology has been reached. It is a *life-long process of continuous encoding* the genotype into a phenotype.[11]

There is one important aspect that has to be mentioned. The *co-evolution of robot body and robot brain*. The metamorphosis and mesomorphosis can be done by *one*

[11] An example might be clarifying. Suppose a morphodynamic controller that implements a game-of-life-like *snake*. This is a *plastic* snake. It continuously adds a robot module to the head of the snake, and removes a module from the tail. The process of mesomorphosis now needs to turn on appropriate sensors in the new head module, connect it to existing body modules and disconnect the virtual "sensor fusion wires" to the detached tail module.

Table 3.3 This table is an overview of different encoding schemes for genetic algorithms. The term compressibility is meant in a computer science sense. The term topology is meant in a network topology sense in which phenotypes can be presented as a directed graph of interacting components. In this context, those components are sensors, sensor fusion filters or associative memory modules.

Methodology	**Corresponding scope**
Direct encoding without self-organisation	There are no large scale, coordinated, changes in network topology from robot to its offspring possible. There is no compression of the topology;
Indirect encoding without self-organisation	Large scale, coordinated changes in network topology across generations are possible. There is *compressibility* of the topology. The indirect encoding mechanism itself does not provide for eventual post-developmental dynamics.
Indirect encoding with self-organisation (evo-devo)	Large scale, coordinated changes in network topology across generations are possible. There is compressibility of the topology. *Post-developmental* dynamics is possible, but not necessarily implemented.
Life-long indirect encoding (eco-devo)	Large scale, coordinated changes in network topology across generations are possible. There is compressibility of the topology. Post-developmental dynamics is implemented. The *environment* can influence development and post-development (metamorphosis).

and the same gene regulatory network. This is done by Bongard for morphogenesis in "Making Evolution an Offer it Can't Refuse: Morphology and the Extradimensional Bypass" (Bongard & Paul, 2001). The sensor fusion model is not allowed to exert control. However, the sensor fusion can be *interfaced* very easily with *morphodynamic control* if the latter also uses a gene regulatory network.

A summary of the layers in the sensor fusion model:

- The sensory-motor layer, see (Tani, 2005);
- The sensor fusion layer, see (Itti *et al.*, 1998), (Grossberg, 1987) and (Baars, 1988);
- The eco-devo layer, see (Bongard & Paul, 2001) and (Quick *et al.*, 2003).

Those individual models all lack certain functionality. However, in its entirety, the three layers implement a sensor fusion model that respects sensory-motor and sensor fusion categorisation, as well as cognitive capabilities like attention, association, anticipation and awareness.

3.3 Application of Embodied Cognition to the Development of Artificial Organisms

Paul Levi

Cognition accepts many definitions depending on the various disciplines it embraces like philosophy, psychology, artificial intelligence, robotics, etc. Let us consider the overall, general-purpose definition proposed by an authority as the Encyclopedia Britannica: "cognition includes all processes of consciousness by which knowledge is accumulated, such as perceiving, recognizing, conceiving, and reasoning. It is one of the only words that refers to the brain as well as to the mind." Put differently, cognition in general view is the set of all processes of thinking (thoughts) and the ability to generate high level patterns of behaviour and interaction (communication) in a society where other thinking creatures act or to adapt to the restrictions of the inanimated (lifeless) nature. In this chapter we focus on a more restrictive definition of cognition that is relevant to the distributive perspective, including inference technologies, planning, decision-making, learning, adaptability, context-awareness, and communication (negotiation).

This chapter will first investigate models of self-organisation and adaptability in inanimated nature relying e.g. on diffusion, Brownian motion, and impacts. Here all such processes are considered as usually force-based that can be described as implicit communications. This was the historical way how the phenomena of self-organisation and collectivity have been detected and modeled e.g. by the methods of synergetics or by the techniques of dynamical systems. An example of the synergetic approach is given by the continuous selection equations. An additional example is presented by the discrete and numerical unstable process of grazing impact.

In the next step we use in addition probabilistic methods in order to define in a formal way the basic concept of information and knowledge. We apply this definition on the computation of the knowledge of the communication complexity during the ongoing compatibility check of robot cells that are involved in the development of an organism (body phase).

All methods of inanimated nature can also be applied to living systems. The significant difference between animated nature and inanimated nature is the ability of living beings to be cognitive and therefore e.g. gather information, evaluate this information and perform autonomous decisions. In the second part we extend the methods of inanimated nature of the first part to cognitive agents that are able to check the compatibility of their knowledge base, their functionalities, their behaviour in order to develop an organism and to control the sustainability of this organism. The merits of this second approach will be demonstrated in more detail by this assembling and surviving process of an organism. The basis for this application is the assumption that creation of living beings is from the very beginning the collection and appropriate managing of information. This means that cognitive processes are not only a matter of an already developed brain and body but are also active during the development of an organism.

3.3.1 *Natural vs. Artificial Systems: Collectivity and Adaptability in Inanimated Nature*

Many natural phenomena of inanimated nature, such as clouds, rainbows, and sunrises are considered as self-organised processes. Some of them are shown in Fig. 3.19. Self-organisational phenomena that have been observed in laboratories are for example Bénard-Cells, the Belousov-Zhabotinsky-reaction, and the operation of a laser. Our goal in this section, focusing more on the laser example (Haken, 1983), is to propose a general model accounting for collectivity and adaptability in lifeless environments.

A main difference compared to most physical approaches is to consider the system under consideration as an open one, meaning that we have to consider dissipation effects and fluctuating forces representing the environmental influences. This difference is characteristic of self-organising processes and makes it possible to account for the creation and annihilation of information (decrease of entropy), whereas closed systems can only see their entropy increasing after the 2nd law of thermodynamics. Fig. 3.20 summarizes the self-organisation in the synergetic view (Haken, 1988).

The meaning of an order parameter and a control parameter can be clearly described in the example of a laser. Control parameter: this parameter is defined by the laser threshold. It is an artful combination of relaxation time and the proportion of spontaneous emissions to thermal emissions. Order parameter: this corresponds to the amplitude of the electromagnetic field. Behaviour of the field : beneath the threshold the laser operates like a lamp (no synchronisation) and the order parameter is connected to a high damping constant that circumvents its influence to the atoms (no real order operations). Above the threshold the field gets coherent and slaves all atoms to synchronise their photon emission. The behaviour within the threshold is characterised by the emergence of a bifurcation of the control parameter.

More generally spoken, a control parameter connects typical constraints like temperature, humidity, pressure, conductivity, vector fields etc. into a relevant combination. A further famous control parameter for fluids is the Reynold number R.

Fig. 3.19 Self-organisation processes in inanimated nature (source: Wikipedia).

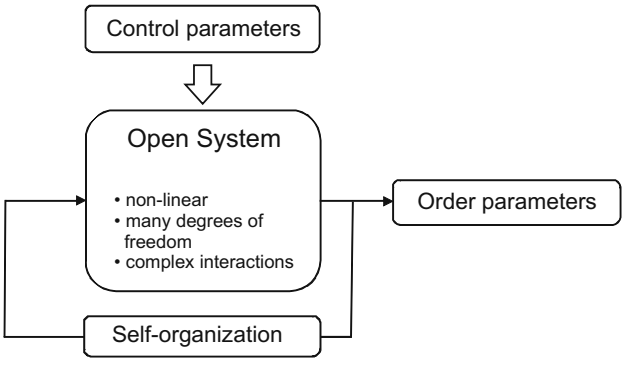


Fig. 3.20 Synergetic representation of self-organisation.

A basic equation of self-organisation, rested upon on Quantum Field Theory, has been presented in Sec. 1.2 and will be here repeated here:

$$\frac{d\widehat{b}}{dt} = c\widehat{b} - d(\widehat{b}\dagger\widehat{b})b + \widehat{F}_{fluct}. \tag{3.1}$$

Here $\widehat{b}\dagger$ and $\widehat{b}$ are bosonic creation respectively annihilation operators. The operator $\widehat{b}\dagger$ describes an amplitude of an order parameter field (better naming would be order operator field), c plays the role of a control parameter (primarily describing damping), d is a constant and $\widehat{F}_{fluct}$ represents fluctuating forces exerted by the environment. These forces can be described by the fluctuation of position and phase of a fictive particle that correspond to Brownian motion and diffusion in classical systems. In a non-physical mathematical way fluctuations are unavoidable, since only with them the commutation rules between the bosonic operators can be exactly fulfilled for all times.

Typical to this non-linear differential equation is that they define the temporal solution of an operator and not, as usual, that of variables or parameters; and that also for quantizing operators exist self-organising differential equations that define coherent states of fields (e.g. multi-mode light fields). Characteristic of the solutions of 3.1 is that they find (without additional strong perturbations) only solutions that are elements of an invariant set of fixed points representing asymptotic system stability.

At the latest we have now to explain to the reader why we mention this quantum mechanical description here. There are two main reasons. Firstly, we assume that the dynamic of regulatory biological networks of genes or cell networks can be modeled by quantization effects since in these dimensions such effects occur for sure. In our application field we are also forced in this direction if we are able later on to develop organisms that are e.g. based on the NEMS (Nano Electronic Mechanical System) technology or even on molecular level. Secondly, we assume that information in general is a quantized term that can be defined by an appropriate general operator. The specialized information is given by the expectation values of this

operator describing its application in a dedicated context. In the logical consequence this means that we replace in our thinking e.g. quantized light field by quantized message fields.

Later on we will present the so called selection equations; these are also self-organising equations but for order parameters of macroscopic systems (no operator equations) that converge only to fixed points ("winner takes it all") depending from the initial values.

Irrespective of the equation that will be solved, it has to be pointed out that the solutions show asymptotic stability (resistance to perturbations) and adaptability e.g. as a series of stable pattern formations if conditions change. This fact again demonstrates that the methods of non-linear systems are in both cases well suited to describe the states of systems and their deviations if restrictions change.

3.3.1.1 Collectivity in Inanimated Nature

Let us contrast a complex natural phenomenon like water and clouds with an artificial one, the laser. We can observe in both cases a standard pattern of self-organisation as it has been presented in Fig. 3.20. All these systems are controlled by an external control parameter, respectively a combination of temperature, humidity, and pressure for cloud (water). This control parameter commands the aggregation of elements (here, H_2 molecules). This cluster ("collectivity") is called water. This aggregation might be unstable, resulting in a "flickering clustering" effect, responsible for phenomenas like such as the water surface tension and the building of dynamic clusters. What matters in the perspective of a complex natural system is the dynamics (self-organisation) of the element clusters; elements go through the so called water-cycle, alternating evaporation (clouds) and condensation (rain)[12].

The control parameter for a laser is defined by a combination of the damping constant of the light field, the life span of an atom, the unsaturated inversion, etc. It represents a threshold, beneath it the light field is not coherent, above it the light field is coherent and operates as a laser. In analogy to the water-cycle we observe a laser-cycle that describes the interactions between the atoms e.g. of a resonator and the light field. The atoms of the resonator (correspond to water) generate a light field (corresponds to vapour) that consists of individual quanta (corresponds to vapour droplets); this field hits on atoms of the resonator and generates again a light field by absorption, enforcing the field, and so on. If we go deeper into details, we observe a self-organising process that generates in addition a coherent light field; its base is the feedback between the field and the atoms that initiate the correlated

[12] This means the existence of H_2O molecules that only exist if there are covalent bindings between two H atoms and one Oxygen atom (typical length of an O-H binding is about 197 pm) can be established (interactions of the electron shells). Whether this connection is stable or not is dominantly depending on the existing temperature. Up to approximately 100°C (sea level) the water molecule is stable, above it will be vaporized. The aggregation is possible since these molecules have electrical dipole moments that generate the connection between these molecules (hydrogen bridges).

stimulated emission, where the field acts as an internal regulative parameter (order parameter) on the atoms (enslaving principle).

After this parallel, three key properties of the system under investigation must be emphasized in order to point out the differences of the methods that are applied in inanimated nature and animated nature to build collectives:

1. CONTROL. The trigger to start the control of a process is usually set externally (control parameter); the environmental conditions allow the activation of a self-organising process (e.g. temperature for molecules). If such a process is activated then internal control (order parameter) stabilises and monitors this process. Cognitive agents operate similarly but their increased abilities allow the utilisation of more complex regulative methods. But the dominant difference between natural creature and processes of the inanimated nature is the fact that living beings control information explicitly, whereas non living control information only implicitly. Another difference is the fact that in inanimated nature the participating units (e.g. molecules) are pure passive and are not autonomous like agents that are active by themselves as e.g. living beings.
2. COMMUNICATION/INTERACTION. The methods of inanimated nature are usually restricted to fluctuations, diffusion and Brownian motion. If coupled differential equations are solved then partial solutions between the participating computers are exchanged, but this exchange is not comparable to negotiations between autonomous agents that are using a high level language. But in the sense of embodied cognition communication is not only focused on this abstract level. Chemical/physical effects that generate smells, perfumes, and the like have also to be considered as a very elementary form of communication. A striking example for this statement are pheromones (example of prophylaxis) that are used by bees. Basic ingredients of such chemical/physical processes are again e.g. fluctuations and diffusion, therefore we cannot state convincingly that only an information exchange in a high level language is a communication and the interactions by smells, forces, etc. is no real communication. This strong distinction is brittle; the transition from low level communication to high level communication is fluent[13].
3. DECISION. The ability of an autonomous agent to make a decision that is more than just to say Yes or No is a concise feature of living beings since such an agent can perform cognitive processes like articulation in a language, inference capabilities, planning capabilities, context awareness, etc. On the other hand, the reactions e.g. of water molecules that depend also on the spin configurations of the participating atoms can be considered as an elementary form of a Yes or No decision. Thus also the ability of decision is not a clear feature of animated nature, since the difference with spin of atoms is not that striking.

Especially the third point explains why in a not animated nature explicit (conscious) justifications to form cooperative groups that are valid for cognitive

[13] In inanimated nature the coupling between two passive units is usually generated only by chemical/physical attractive forces if the units are close enough (e.g. distance of some hundred pm's for H_2O molecules).

agents - like increase of performance, scalability of performance, increase of stability (robustness), increase of reliability etc. - cannot be given by the participating "particles". However the methods that have been developed for the modeling of processes of inanimated nature can also be applied to biological systems as e.g. has been demonstrated by (Meinhardt, 1995). For us it is important to note that such methods can also be deployed for the generation of creatures as an integrative and additive part of cognitive agent abilities. The essential point is that such an agent can learn which procedure is the best one and how to implement its brain (body).

The inhibitor and activator model of Meinhardt describes the external pattern formations of sea shells by the interactions of different concentrations by the way of methods of chemical reactions. But his wording has to be interpreted in the sense of morphogenesis and not in the sense of cognitive agents. The differential equations defined by him are:

$$\frac{\partial a}{\partial t} = s(\frac{a^2}{b} + b_a) - r_a a + D_a \frac{\partial^2 a}{\partial x^2} \tag{3.2}$$

$$\frac{\partial b}{\partial t} = sa^2 - r_b b + D_b \frac{\partial^2 b}{\partial x^2} + b_b. \tag{3.3}$$

Here are $a(x)$ the autocatalytic activator and $b(x)$ the inhibitor. D_a and D_b are the diffusion coefficient, and r_a and r_b are the decay rate of a and b. Fig. 3.21 shows the result of this method to decorate sea shells.

Fig. 3.21 The robe of a sea shell computed by the Eqs. (3.2) and (3.3).

3.3.1.2 Adaptability in Inanimated Nature

The description of adaptability (ability to adaptation) of non creature is at best given by aggregation states of a H_2O cluster. We differentiate the following phases: solid (ice), liquid (water) and vaporous (steam). The aggregated states of such molecule clusters are primarily controlled by the external conditions temperature and pressure. Ice has a rigid lattice structure (e.g. crystalline hexagonal (snowflake) structures, Ice I_h; or cubic structures, Ice I_c) since the hydrogen bonds are fixed by more bonds between the oxygen atoms than in a liquid state. The hydrogen bonds are forced to open if the temperature is higher than e.g. 100°C (vapour). These phase transitions describe a perfect adaptation to external thermal (pressure, temperature)

restrictions. These state transitions have been described so far by the usual and more understandable classical physic-chemical explanations.

For our field of applications we prefer a more mathematical description of adaptability in the sense of nonlinear dynamic systems (Guckenheimer & Holmes, 1983), (Mikhailov & Calenbuhr, 2002). Here our first specification is devoted to fixed points (equilibrium points) that are stable or asymptotically stable. In more detail this means that we are looking for differential that generates solutions that are elements of an invariant set of fixed points. Small perturbations (e.g. impact of a water drop on a water surface, small deviations of components in a rigid body caused by stress) do not disrupt the whole system but there can be observed a pull back from the perturbations to the original or another fixed point of the invariant set of equilibrium states. A second example of such an adaptive behaviour is the disruption of input data (e.g. distorted images) that does not change the general solution (recognition of distorted images). A third example is the disruption of some coupling constants between interacting units that are dynamically processed by short time deviations but end up again, in a maybe different, stable solution.

If the external or initial conditions dominantly change then adaptability implies that a series of different pattern formations (e.g. different aggregations of H_2O clusters, or different morphological forms of artificial organisms) as a series of bifurcations starts without ending in chaotic states. An elegant method to describe such a bifurcation series is the symbolic dynamic (Avrutin, 2009).

We now start with the description of two different differential equations in order to demonstrate what kind of mathematical effects are generated if collectivity or/and adaptability are demanded. We start with the selection equations that are followed by the map for a grazing impact.

Selection Equations

There already exist different methods to solve assignment problems (e.g. Hungarian method), but we choose here the selection equations since they guarantee a unique assignment between two sets (2-index problem) in real time. A possible disadvantage of this method is the fact that no guarantee can be provided that the global maximum has been selected. The original selection equation is an equation for order parameters (modes) ξ_i and is defined by

$$\frac{d}{dt}\xi_i = \kappa\xi_i(1 - \xi_i^2 - \beta \sum_{i' \neq i} \xi_{i'}^2) \tag{3.4}$$

If the initial value for ξ_i is greater than zero, and $\beta > 1$, then the mode with the greatest initial value converges to 1 and all other modes "die out". The parameter κ will be used to accelerate or slowdown the calculation. In distributed computations of the ξ_i's the different modes exchange mutually the terms $\beta \sum_{i' \neq i} \xi_{i'}^2$ during the iterating process of problem solving.

The operation mode of this equation is at best represented by Fig. 3.22. It demonstrates the recognition of a horse (emblem of Stuttgart) from a distorted (noisy) input

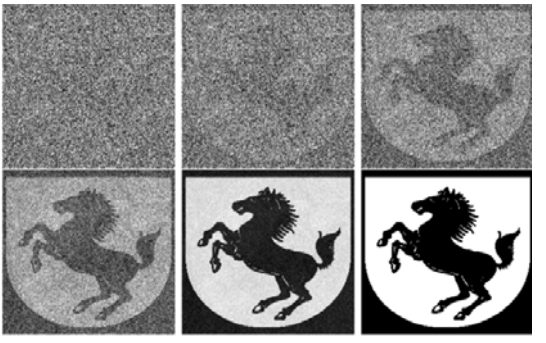


Fig. 3.22 Adaptability of the original selection equations to associate a corrupted input image with a correct reconstruction of the original object.

image in some pattern formation steps where each mode represents a different position of this horse and the mode that represents the correct orientation of the horse is the sole winner (Haken, 1988). Each mode ξ_i represents a different attractor in a configuration space (possible states of all modes) with different basins of attraction. The structure of such a space can be descriptively represented in the adequate potential field, the attractors are the minima and the symmetry breaking ridges are the borders of the basins of attractions. Such a space is also named associative memory. Further it should be mentioned that the adaptation abilities of the Eq. (3.2) are very similar to that one of Kohonen nets.

The standard selection Eq. (3.2) can also be considered as a method to solve the 2-index assignment problem of combinatorial optimization. If the two sets of the assignment problem have the same cardinality then each mode can be considered as permutation matrix that describes different patterns. The original equation can be extended to an assignment matrix (ξ_{ij})-element i from the first set is assigned to j of the second set - and by two additional, time dependent parameters $\alpha(t)$ and $\beta(t)$ in order to increase adaptability and robustness of this approach (Lafrenz *et al.*, 2008). The adaptability means that the assignment is performed dynamically (on line change of assignment if e.g. initial conditions change). The increased robustness assures the correctness of the solution even if some of the exchanged partial results are distorted. The modified coupled selections equations have the form:

$$\frac{d}{dt}\xi_{ij} = \kappa\xi_i(\alpha(t)_{ij}(\lambda_{ij}(t) - \xi_{ij}^2) - 2\beta\xi_{ij}^2 - \beta(\sum_{i'\neq i}\xi_{i'j}^2 + \sum_{j'\neq j}\xi_{ij'}^2) \qquad (3.5)$$

Here $\alpha(t)_{ij}$ are context based utility functions and $\xi_{ij}(t) \in \{-1,1\}$ are switching parameters to activate or deactivate assignment processes in order to perform the mapping stepwise between selected subsets of the original sets.

Grazing Impact

Impacts with zero velocity are called grazing impacts (Molenaar *et al.*, 2001). In this special case the usually continuous description of the collision of a harmonic

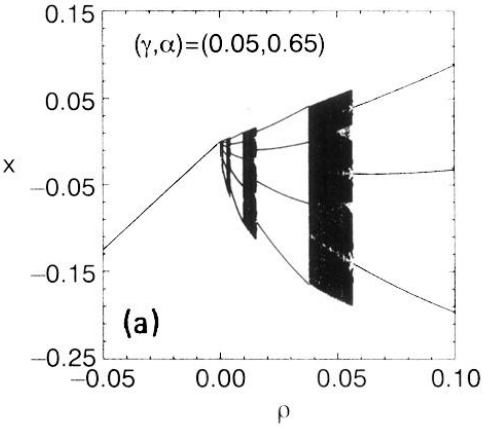


Fig. 3.23 Bifurcation diagram ($\tau^2 = 1$) of the Nordmark map. For $\rho < 0$ there is an attracting period-1 orbit, Eq. (3.6), after the bifurcation at $\rho = 0$ the reverse period incrementing starts, Eq. (3.6).

oscillator that is sinusoidal forced and hits a hard (or compliant) wall with zero velocity can be transferred into discrete equations, so called maps. This step is important since there are two sources of discontinuity in reality: the impact itself and the energy loss by the impact (collisions that are inelastic). In the context of our contribution this is important for e.g. ball bearings, loosely fitting joints, abrasive electrical contacts and for multi-legs robots. The original simplified (hard wall) Nordmark map demonstrates how such a phenomena can be discretized and numerically solved. This map is given by:

$$\begin{cases} x_{n+1} = \alpha x_n + y_n + \rho \\ y_{n+1} = -\gamma x_n \end{cases} for\, x_n \leq 0\, no\, impact, \tag{3.6}$$

$$\begin{cases} x_{n+1} = -\sqrt{x_n} + y_n + \rho \\ y_{n+1} = -\gamma \tau^2 x_n \end{cases} for\, x_n > 0\, impact, \tag{3.7}$$

Here x_n and y_n are the discrete position and velocity calculated at time t_n (discrete phase space), and the most important parameter is the bifurcation parameter ρ ($|\rho| \ll 1, \rho = 0$ (grazing incidence)). The point that is interesting for us is the square-root singularity ($\frac{\partial x_{n+1}}{\partial x_{n+1}} = -\frac{1}{\sqrt{x_n}}$, singular at $x_n = 0$). This special singularity causes the effect of period incrementing that is shown in Fig. 3.23 (remember here the case of period doubling bifurcations of the quadratic nonlinearity considered by Feigenbaum). We observe a reverse period incrementing (window of stable periodic behaviour) that is followed by a chaotic band (Chin *et al.*, 1994).

After we have presented some selected differential equations that primarily have been developed to model processes of natural and artificial processes we continue this procedure and consider probabilistic methods in order to integrate a concept of information and knowledge into our considerations.

3.3.2 *Definition of Information and Knowledge Related to Restrictions*

We clearly have to differentiate between the phase where an organism is created and the phase where an organism "lives" and operates in some environment. In the second phase energy homeostasis, harvesting and cooperation for task performance are essential. In natural living beings these activities are supported by usually unconscious periodic actions like breathing, heartbeat or synchronisation signals in the brain. This second phase will be evaluated by one or more fitness functions.

For us information is a concept that does not stand alone as it is usually defined but it has to be formed in a strong connection with the restrictions that are to be satisfied in the actual context (context-awareness). Therefore we have at least to distinguish the boundary conditions for the two mentioned phases. Nevertheless there exist methods to calculate a probabilistic concept of information by general probability constraints (e.g. expected number of messages, expected energy expectation, the compliance of Kullback's minimum cross entropy principle, or Jayne's maximum entropy principle entropy (Kapur & Kesavan, 1992)).

We use the notion of information in the sense of the concept of generalised entropy, meaning that we stand away from the original thermodynamically defined concept (e.g. order, disorder of a closed or open system) but enlarge it in direction of uncertainty, equality, diversity, similarity, randomness, etc. Therefore if we speak of information we mean all the time the "augmented" concept of entropy that also can be characterised as information-based entropy.

In this chapter we demonstrate how the information-based entropy is usually calculated in probabilistic approaches, in physics and in physical chemistry. The main missing part in all of these mathematics-based methods is the concept of learning and deciding which is typical for cognitive approaches. After we present these classical methods we enlarge our proceeding in the fourth chapter by cognitive agent technology as a clear enlargement of pure mathematical procedures.

3.3.2.1 Information Related to Probabilistic Restrictions

We begin our discussion by the proposal to measure information by information-based entropy that in addition has to fulfill a set of constraints. Here we consider first the restrictions of probability distributions have to satisfy. We start again (see Sect. 1.2) with a simplified (without field operators) state sum for the total state of agent j:

$$S_j = \sum_k e^{(\mu_j(<N_{jk}^2>-<N_{jk}>^2)-W_{jk})/<N_j^2>)}, \tag{3.8}$$

where $< N_{jk} >$ is the mean value of messages that agent j gets from agent k and $< N_j^2 >$ is the expectation value of the squared number of all messages agent j has been get. This magnitude is selected by the correspondence to the Boltzmann term kT.

In the first category of constraints we focus ourselves to probability constraints. We describe the knowledge state distribution W_{jk} by a symmetric Kullback-Leibler cross-entropy that calculates the difference of two probabilities:

$$W_{jk} = J(\underline{p}_{(jk)} : \underline{q}_j) = \sum_i p_{(jk)i} \ln \frac{p_{(jk)i}}{q_{ji}} + \sum_i q_{ji} \ln \frac{q_{ji}}{p_{(jk)i}}. \tag{3.9}$$

Here $p_{(jk)i}$ describes the wanted probability that agent j transfers in the internal state i of r states if it gets a message from agent k; and q_{ji} describe the a priori probability that an agent j is in the knowledge state i. This expression must be minimised by the following restrictions:

$$\sum_{j=1}^{n} p_{(jk)i} = 1, \sum_{j=1}^{n} q_{ji} = 1, i = 1, ..., r. \tag{3.10}$$

$$\sum_{i=1}^{r} p_{(jk)i} g_s(x_i) = a_s, s = 1, 2, ..., m. \tag{3.11}$$

A typical example for the side condition (3.11) is:

$$\sum_{i=1}^{r} \sum_{j=1}^{n} j p_{(jk)i} = \sum_{i=1}^{r} < N_{ki} > = < N_k >, \tag{3.12}$$

this is the expected number of agents that changes in a new knowledge state if agent k sends to all agents the same message. Another restriction is the expected total energy consumption for all state transitions.

$$\sum_{1=1}^{r} E_i \sum_{j=1}^{n} j p_{(jk)i} = < E > \tag{3.13}$$

With the aid of Lagrange multipliers $\lambda_i, i = 0, ..., m$, and the restrictions for the $\underline{p}_{(jk)}$ distribution we get the general result for this distribution (Kapur & Kesavan, 1992):

$$\underline{p}_{(jk)i} = q_{ji} exp(\lambda_0 - \lambda_1 g_1(x_i) - \lambda_2 g_2(x_i) - ... - \lambda_m g_m(x_i)). \tag{3.14}$$

For example the Poisson distribution

$$q_{ji} = e^{-b} \frac{b^i}{i!} \tag{3.15}$$

with the expectation value b and the mean of the $\underline{p}_{(jk)}$ distribution is prescribed as m then the resulting distribution is

$$p_{(jk)i} = e^{-m} m^i / i!, i = 0, 1, 2, . \tag{3.16}$$

The fact that this distribution is again Poisson with the claimed mean m demonstrates that the form of $p_{(jk)i}$ is dictated by the a priori distribution q_{ji} while the mean m is independent from the mean b. The result for W_{jk} is as follows

$$\begin{aligned} W_{jk} &= (\sum_i e^{-m}\frac{m^i}{i!}\ln(e^{-m}\frac{m^i}{i!}/e^{-b}\frac{b^i}{i!}) + \sum_i e^{-b}\frac{b^i}{i!}\ln(e^{-b}\frac{b^i}{i!}/e^{-b}\frac{b^i}{i!}) \\ &= \sum_i (b-m+i(\ln(m)-\ln(b))[p_{(jk)i}-q_{ji}]. \end{aligned} \tag{3.17}$$

In the next step we calculate the statistical potential

$$\Omega_j = -<N_j^2>\ln S_j. \tag{3.18}$$

Afterwards agent j holds the information:

$$\begin{aligned} &Inf_j = -\frac{\partial\Omega_j}{\partial(<N_j^2>)} = -\ln S_j + \frac{1}{<N_j^2>} \\ &\sum_k \frac{(\mu_j(<N_{jk}^2> - <N_{jk}>^2) - W_{jk})e^{(\mu_j(<N_{jk}^2>-<N_{jk}>^2)-W_{jk})/<N_j^2>}}{S_j} = \\ &-lnS_j + \frac{1}{<N_j^2>}\sum_k \mu_j(<N_{jk}^2> - <N_{jk}>^2)\widetilde{p}_{jk} \end{aligned} \tag{3.19}$$

Here

$$\tilde{p}_{jk} = \frac{e^{\mu_j(<N_{jk}^2>-<N_{jk}>^2)-W_{jk}/<N_j^2>}}{S_j} \tag{3.20}$$

represents the standard probability definition of a state sum.

For the special case that all distributions are Poisson with different means we can calculate (3.19). For S_j we assume that the individual probabilistic distribution for the message exchange has mean c_{jk} and the total probabilistic distribution for the message exchange has mean c_j. It follows

$$<N_{jk}^2> - <N_{jk}>^2 = c_{jk} \text{ and } <N_j^2> = c_j(c_j+1), \text{ and } S_j \tag{3.21}$$

can now expressed as

$$S_j = \sum_k e^{(\mu_j c_{jk}-W_{jk})/c_j(c_j+1)}. \tag{3.22}$$

The total information that agent j can calculate till now is

$$\begin{aligned} Inf_j &= -\ln S_j + \frac{1}{c_j(c_j+1)}\sum_k \frac{(\mu_j c_{jk}-W_{jk})e^{(\mu_j c_{jk}-W_{jk})/c_j(c_j+1)}}{S_j} \\ &= -\ln S_j + \frac{1}{c_j(c_j+1)}\sum_k (\mu_j c_{jk}-W_{jk})\widetilde{p}_{jk}, \end{aligned} \tag{3.23}$$

with the Boltzmann like probability

$$\widetilde{p}_{jk} = \frac{e^{(\mu_j c_{jk} - W_{jk})/c_j(c_j+1)}}{S_j}. \tag{3.24}$$

What is still missing in formula (3.23) is the definition of the "chemical" potential. We set it here to $\mu_j = \mu_j^0 + \ln(activity_j)$, whereas μ_j^0 is the standard potential and the activity is calculated with respect to a standard activity (normalized activity). By the direct indication what kind of activity will be described, e.g. interaction with other agents (substances), phase transition or spatial diffusion detail, the dependencies of μ_j respectively $activity_j$ from other magnitudes must be modeled separately for each different context. In chemistry the chemical potential defines the direction of diffusion between two chemical substances (e.g. fluid, gaseous). If the chemical potential is equal for both sides of the chemical reaction then the inter mixture stops and equilibrium employs. We use the chemical potential e.g. in an elementary way by

$$\mu_j = < N_j^0 > + \ln \frac{< N_{jk} >}{< N_j^2 >}. \tag{3.25}$$

Here $< N_j^0 >$ is the standard expectation value of the number of messages that agent j sends out to all other agents, the means in the quotient have been defined above. So in the end one possible elementary formula for the information (information based entropy) for agent j is:

$$Inf_j = -\ln \sum_k e^{((<N_j^0> + \ln(\frac{<N_{jk}>}{<N_j^2>}))c_{jk} - W_{jk})/c_j(c_j+1)}$$
$$+ \frac{1}{c_j(c_j+1)} \sum_k \frac{((< N_j^0 > + \ln \frac{<N_{jk}>}{<N_j^2>})c_{jk} - W_{jk}) e^{((<N_j^0> + \ln(\frac{<N_{jk}>}{<N_j^2>}))c_{jk} - W_{jk})/c_j(c_j+1)}}{S_j}, \tag{3.26}$$

where S_j is given by the argument sum within the ln expression. We have observed that in our definition of information the knowledge W_{jk} defined in (3.17), is a basic and important ingredient. We calculate this term by a measure of symmetric divergence. In the following two sub-chapters we calculate first the discrete probabilistic knowledge for the compatibility test of individual components by constructing an artificial organism. Hereby we consider the individual components as agents that send themselves mutually their individual information in order to prove the compatibility of two components that are able to build a new compound, stable component. Afterwards we calculate the structural stability of a finished organism by a continuous knowledge representation of the probability of failure.

3.3.2.2 Information Related to Creating Restrictions (Construction Problem)

Let C_j be the number of components (robot cells) of type j, where $j = 1, ..., n$, defining n different types of basic bricks. Let O_k be the number of all operations

that can be performed on component k in order to establish a pair wise stable and functional successful connection between all components j and the individual component k, $k = 1,...,n$. If $O_k = 0$ then no connection can be established at all. Let M_{jk} be the number of message exchanges of pair wise successful information transfers between component j and component k, where both parts are considered as synchronized agents. The result of these information exchanges correspond to a compatibility check between component j and component k. The task we have to perform is now to estimate the $n^2 M_{jk}$'s as individual probabilities in order to get an assessment of the communication complexity (modified transportation problem). The natural constraints are:

$$\sum_{k=1}^{n} C_k = C_j, j = 1,2,...,n, \tag{3.27}$$

$$\sum_{j=1}^{n} M_{jk} = O_k, k = 1,2,...,n, \tag{3.28}$$

$$\sum_{k=1}^{n}\sum_{j=1}^{n} M_{jk} c_{jk} = \widetilde{c}. \tag{3.29}$$

Constraint (3.27) reduces the number of messages that agent (component) j can send to other agents k by the number of available components of type j. Constraint (3.28) restricts the number of messages that all components j can send to component k by the total number of operations that can be performed with component k. Constraint (3.29) gives a measure of the message transfer costs c_{jk}.

From the two constraints (3.27) and (3.28) we deduce a constant M:

$$\sum_{k=1}^{n}\sum_{j=1}^{n} M_{jk} = \sum_{k=1}^{n} O_k = \sum_{j=1}^{n} C_j = M. \tag{3.30}$$

The constant M will be used to interpret the proportions M_{jk}/M as probabilities. We minimise the symmetric Kullback-Leibler measure

$$D(\frac{M_{jk}}{M} : \frac{C_j O_k}{M}) + D(\frac{C_j O_k}{M} : \frac{M_{jk}}{M}) = \sum_{k=1}^{n}\sum_{j=1}^{n} \frac{M_{jk}}{M} \ln(\frac{M_{jk}/M}{C_j O_k/M}) +$$
$$\sum_{k=1}^{n}\sum_{j=1}^{n} \frac{C_j O_k}{M} \ln(\frac{C_j O_k/M}{M_{jk}/M}), \tag{3.31}$$

under the consideration of the constraints (3.27) - (3.29). This gives

$$M_{jk} = A_j B_k C_j O_k e^{-c_{jk}}, \tag{3.32}$$

where A_j, B_k, and λ have to be determined by using the three constraints (3.27) - (3.29). If the cost constraint (3.27) is neglected ($\lambda = 0$), the result is

$$M_{jk} = C_j O_k. \tag{3.33}$$

Both M_{jk}'s divided by$M$ are the two different probability distributions. If we apply the last two formulas to calculate the individual knowledge

$$\begin{aligned} W_{jk} &= \frac{M_{jk}}{M} \ln\left(\frac{M_{jk}/M}{C_j O_k/M}\right) + \frac{C_j O_k}{M} \ln\left(\frac{C_j O_k/M}{M_{jk}/M}\right) \\ &= \frac{M_{jk}}{M}[\ln(A_j B_k) - \lambda c_{jk}] - \frac{C_j O_k}{M}[\ln(A_j B_k) + \lambda c_{jk}] \\ &= \ln(A_j B_k)[M_{jk} - C_j O_k] - \frac{\lambda c_{jk}}{M}[M_{jk} + C_j O_k]. \end{aligned} \tag{3.34}$$

The total knowledge W is

$$W = \sum_{k=1}^{n} \sum_{j=1}^{n} W_{jk}. \tag{3.35}$$

This approach can be improved a little bit if we involve the knowledge of the results of previous assemblies M_0 and $M_j k0$, then it is more appropriate to consider the term

$$W_{jk0} = \frac{M_{jk}}{M} \ln\left(\frac{M_{jk}/M}{M_{jk0}/M_0}\right) + \frac{M_{jk0}}{M_0} \ln\left(\frac{M_{jk0}/M_0}{M_{jk}/M}\right). \tag{3.36}$$

This calculation delivers a better estimation.

3.3.2.3 Information Related to Structural Restrictions

The classical physical method to calculate equilibrium condition of deformable rigid bodies with a fixed structure is to construct a stress tensor $S_{\alpha\beta}$ $(\alpha, \beta = 1, ..., 3)$ that can be used to calculate for each normal direction a stress vector, and in addition to evaluate a dilatation tensor $D_{\alpha\beta}$ $(\alpha, \beta = 1, ..., 3)$ that describe the kind of deformation (e.g. strain, torsion) if external forces affect the body. By the use of the elastic potential U it is possible to establish the stress-strain connection. The elastic potential can further be engaged to calculate the final basic integral equations for equilibrium conditions (resultant law and momentum law). We now consider the stability of a completed artificial body that is rigid by a continuous probability calculation of failure. Let $p_f(x)$ be the probability density that a component fails if it is subjected to a stress x, where $(p_F(x)dx)$ is the probability that a component fails if it subjected to a stress between $x \pm dx$). What we search for is the estimation of this probability density and we do it by formulation of an adequate constraint. The distribution function of failure (cumulative probability) is as usual defined by

$$P_F(x) = \int_0^x p_F(x)dx, \tag{3.37}$$

it determines the probability of failure if a component is subjected to a stress up to x. From a fraction of components $p_s(x)dx$ that are experimentally subjected to

stress up to x the expectation value for the distribution function P_{fail} is known. This expectation value of $P_F(x)$ is given by

$$P_{fail} = \int_0^X P_F(x) p_s(x) dx. \tag{3.38}$$

All components fail (brake) definitely if they are subjected to the maximal stress X. Integration of (3.38) by parts delivers the result

$$P_{fail} = [P_s(x) P_F(x)]_0^X - \int_0^X P_s(x) p_F(x) dx. \tag{3.39}$$

Now it is valid that $P_s(X) = 1, P_F(X) = 1, P_s(0) = 0, P_F(0) = 0$, so that we get the wanted constraint for $p_F(x)$

$$1 - P_{fail} = \int_0^X P_s(x) p_F(x) dx. \tag{3.40}$$

The second constraint is the unavoidable standard constraint

$$\int_0^X p_F(x) dx = 1. \tag{3.41}$$

If we use Jaynes' maximum entropy principle then we maximize

$$-\int_0^X p_F(x) \ln p_F(x) dx \tag{3.42}$$

The Lagrange function L subjected to the two constraints with additional two multipliers λ_0 and λ_1 looks like

$$\begin{aligned} L = &-\int_0^X p_F(x) \ln p_F(x) dx - (\lambda_0 - 1)(\int_0^X p_F(x) dx - 1) - \\ &\lambda_1 (\int_0^X P_s(x) p_F(x) dx - 1 + P_f ail). \end{aligned} \tag{3.43}$$

If we set the derivative of L with respect to p_F equal to zero we get

$$-1 - \ln p_F(x) - (\lambda_0 - 1) - \lambda_1 P_s(x) = 0. \tag{3.44}$$

From this equation we deduce the result

$$p_F(x) = e^{-\lambda_0 - \lambda_1 P_s(x)} = \frac{e^{-\lambda_1 P_s(x)}}{\int_0^X e^{-\lambda_1 P_s(x)} dx}, \tag{3.45}$$

where λ_1 can be determined by inserting this result into constraint (3.40). The final distribution density $p_F(x)$ is a continuous variate version of the Maxwell distribution. This was the classical calculation of $p_F(x)$, but closer to our developmental line is the deployment of the continuous symmetric Kulback-Leibler measure (cross

entropy). Here we involve in addition to the failure distribution density $p_F(x)$ a new different failure distribution density $q_F(x)$ together with a new stress probability density $q_s(x)$ and distribution function

$$Q_s(x) = \int_0^x q_s(x)dx, \frac{dQ_s(x)}{dx} = q_s(x). \tag{3.46}$$

The new q distribution density and Q distribution function result from other experimental series. We have therefore to minimize the expression

$$W = J(p_F(x) : q_F(x)) = \int_0^X p_F(x) \ln(\frac{p_F(x)}{q_F(x)})dx + \int_0^X q_F(x) \ln(\frac{q_F(x)}{p_F(x)})dx. \tag{3.47}$$

This expression defines also our total continuous knowledge about the failure of all components that comes from two experimental series.

We begin the evaluation of this expression by the first term, where the constraints stay unchanged and only the Lagrange function includes

$$\int_0^X p_F(x) \ln(\frac{p_F(x)}{q_F(x)})dx \quad instead \quad -\int_0^X p_F(x) \ln p_F(x)dx. \tag{3.48}$$

The result is

$$p_F(x) = q_F(x)e^{-\lambda_0 - \lambda_1 P_s(x)} = q_F(x)\frac{e^{-\lambda_1 P_s(x)}}{\int_0^X e^{-\lambda_1 P_s(x)}dx}. \tag{3.49}$$

For the second term in Eq. (3.47) we obtain

$$q_F(x) = p_F(x)e^{-\lambda_0 - \lambda_1 Q_s(x)} = p_F(x)\frac{e^{-\lambda_1 Q_s(x)}}{\int_0^X e^{-\lambda_1 Q_{s(x)}dx}}. \tag{3.50}$$

After we insert these two equations into (3.47) we get the final formula

$$\begin{aligned} W = {} & q_F(x)\frac{e^{-\lambda_1 P_s(x)}}{\int_0^X e^{-\lambda_1 P_s(x)}dx}[\ln(\frac{q_F(x)}{p_F(x)}) + \lambda_1(Q_s(x) - P_s(x)) + \\ & \ln(\frac{\int_0^X e^{-\lambda_1 Q_s(x)}dx}{\int_0^X e^{-\lambda_1 P_s(x)}dx}] + p_F(x)\frac{e^{-\lambda_1 Q_s(x)}}{\int_0^X e^{-\lambda_1 Q_s(x)}dx}[\ln(\frac{p_F(x)}{q_F(x)}) - \lambda_1(Q_s(x) - P_s(x)) + \\ & \ln(\frac{\int_0^X e^{-\lambda_1 P_s(x)dx}}{\int_0^X e^{-\lambda_1 Q_s(x)}dx})]. \end{aligned} \tag{3.51}$$

This is the total knowledge that we can get from two experimental series. We increase our knowledge if we make more experiments that deliver two further failure probability densities $r_F(x)$ and $s_F(x)$. For convenience we assume that all four probability densities are independent then we can e.g. calculate the non symmetric Kulback-Leibler measure (a joint probability distribution of two distributions p and q we denote by $p * q$):

$$D(p_F(x) * q_F(x) : r_F(x) * s_F(x)) = \int_0^X \int_0^X p_F(x) q_F(x') \ln(\frac{p_F(x) q_F(x')}{r_F(x) s_F(x')}) dx dx' =$$
$$\int_0^X \int_0^X p_F(x) q_F(x') \ln(\frac{p_F(x)}{r_F(x)}) dx dx' + \int_0^X \int_0^X p_F(x) q_F(x') \ln(\frac{q_F(x')}{s_F(x')}) dx dx' =$$
$$\int_0^X p_F(x) \ln(\frac{p_F(x)}{r_F(x}) dx + \int_0^X q_F(x') \ln(\frac{q_F(x')}{s_F(x')}) dx' =$$
$$D(p_F(x) : r_F(x)) + D(q_F(x) : s_F(x)). \tag{3.52}$$

For our knowledge calculation this means that we can just add the individual measures if the probability densities are independent:

$$W = D(p_F(x) : r_F(x)) + D(q_F(x) : s_F(x)) + D(r_F(x) : p_F(x)) + D(s_F(x) : q_F(x)). \tag{3.53}$$

If the distributions are not independent then such formulas are more complex and no more additive, but it is still possible to calculate W. By this way we can go on and append two more new distributions (new experimental series) to the old one in order to increase the knowledge we can get from such experiments. This procedure resembles an elementary learning process. Therefore we stop here with the description of probabilistic methods and focus in the next chapter to agent based cognition (learning).

3.3.3 Collectivity and Adaptability in Animated Nature

The main reason to operate in a team is given if the participating active units (agents) are not able to perform a complex task since none of the participants has the skills and the capacity to solve the problem alone (Ferber, 1999). Further points for collectivity are the increase of performance, scalability and flexibility (Parker, 2008). If the team members are agents that operate on a higher and more abstract level since they posses cognitive abilities, they know explicitly about all possible conflicts like the common goal (e.g. completion of a global task from which they benefit), the individual goals (beliefs, intensions), individual missing capabilities, insufficient resources, etc. So they start to solve the problem by direct negotiations (Weiss, 1999), (Wooldridge, 2002). Very typical for this agent-based approach is the definition of roles (e.g. organiser or planner), the knowledge based performance of roles, the interactions between the agents with the result that the fittest survive (e.g. energy homeostasis), the emergence of "social" behaviour patterns (e.g. egoistic agent, altruistic agents).

The adaptivity inside an organism is more than only the adaptation to parameters. It is also the structural change, the functional differentiation or the creation of regulative rules (see Sect.1.1). For example the cascaded regulation of an organism is at least at the lower of locomotion completely changed if legs of a creature are replaced by wheels.

The basic assumption of this work is that all living beings posses a kind of cognition and can therefore also communicate by some kinds of a "language". This fiction

is obvious true for deliberative agents, but we go further and also guess that this is also true for swarms or even on the component (cell) level. The main difference of these two levels is that on lower levels there is no explicit and conscious deliberation (inference, decision), instead it is completely driven by evolution. This means in our application field e.g. that no team member has a global plan or a complete distributed plan but there exist different, elementary cognitive abilities in the multi-agent system. Independent of the level of organism (high or primitive cognitive abilities), we model all phases of development and adaptability of an organism by agents.

For the creation and the survival of artificial organisms we introduce e.g. evolution-agents (organisers) and fitness-agents as distinguished agents, where both types of agents operate only in common restricted regions (Nüsslein-Volhard, 2004). The whole system can only be generated if these agents exchange information from the beginning among themselves and between evolution-agents and fitness-agents in order to generate an information field that synchronises all together in direction of pairwise compatibility. Such a field operates as an order parameter field. Each agent stores its collected information in its own information-base, and all these local basis together represents a distributed information-base for the whole organism.

We consider the information exchange at each level of creation of an organism as the basic operation that is in nature initiated e.g by physical forces and that is a dominant part of evolution. The result of an evolutionary process e.g. of individual connection of cells is created by a learning process where the result of this process is stored as the sum of separate information values. In this sense the "construction" of a body can be considered as an exploration where at each developmental phase the criteria that have to be learned are different (Fukuda & Ueyama, 1994), (Lungarella *et al.*, 2003). We further assume that the optimisation criteria of evolution is the decrease of information (open systems). Living systems have to be considered as open systems (e.g. dissipation, fluctuations), therefore we consider the process of minimisation of information. This procedure corresponds to the reduction of entropy, whereas we assume that e.g. each state S_j is positive and ≤ 1, see Eq. (3.19).

A fitness function describes the best adaptation of an organism to a task in a given environment, e.g. the ability of locomotion in an environment with existing restrictions can be mandatory for survival. This fact forces, on the other side, the evolution to re-evaluate the so far developed (learned) information value in order to generate a new information that is minimised respectively to the new restrictions. Therefore a fitness function is comparable only in part with a function of reinforcement learning since there is no interaction with the evolution process. The decision which is the best balance between information-based evolution and fitness of an organism will usually be given by the fitness of the body if stringent conditions like survival have to be fulfilled. This is true in the developmental phase.

If the organism is finished and is fit to perform a task then the evolutionary process might define this balance. We will involve evolution-agents and fitness-agents in order to establish the above mentioned balance. This is even e.g. in the lower level of cells true where the fitness-agent has to assure the survival of individual cells and aggregated cells. For higher levels of an organism this might be more obvious where

for example in biological systems the survival is given by the unconscious processes of the organs. Examples are: blood transport by a heart, supply of oxygen by lungs, dialysis of blood by kidneys, etc. All these organs take care that an organism can live, meaning primarily the adaptability and the chance of survival of a creature. We consider therefore the creation of such vital organs from the very first of development since these organs are build from differentiated cells.

3.3.4 Information Based Learning to Develop and Maintain Artificial Organisms

3.3.4.1 Establishing the Individual Information by Mutual State Exchange

We start our consideration from bottom in the sense that each component is represented by an agent that can communicate, and gather information by learning.

This kind of reflection is also inspired by the details of a bio-inspired production process. The combination of two components (cells) is in the first step dictated by the calculation of the individual states defined by (3.8). The knowledge W_{jj} ($k = j$, self-information) is zero in the beginning since the two discrete probability distributions $\underline{p}_{(jj)}$ and $\underline{q}_j$ in Eq. (3.9) are equal, e.g. equal to the uniform distribution $\underline{u}_j = (1/n, 1/n, ..., 1/n)$, where n is the number of different knowledge states.

Further we assume that there exist two different coupling mechanisms for the connection of two components (in the sense of fitting joints) that are refined by additional attributes like strength and angle of joint. Fig. 3.24 demonstrates the two types of mechanical connections we adopt, where only spherical ends bond together or only rectangular ends join (we neglect in the first step electrical connections).

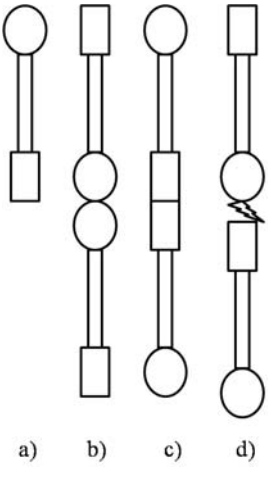


Fig. 3.24 **(a)** Possible forms of components; **(b)** and **(c)** Successful connections, **(d)** Broken joints (thanks to Mr. Oliver Zweigle).

Here it is also very ambitious to fabricate and assemble carbon nanotubes (CNT's), (Fukuda *et al.*, 2007).

Like biological cells do it from the beginning, one component searches in a strong limited environment for other similar components in order to combine with them, we consider this process as the gathering of information that describes whether there can be found another part that fits, or not. This process mimics physical forces that affect the type of connection. In our probabilistic approach of information and knowledge this stands for a knowledge base (better wording would be information base) that an agent starts to fill up with probabilities that a joint is possible or not (since they do not fit, or one component is broken). By this way each component-agent j acquires the knowledge W_{jk} for the component pairs j and k. A fitness agent j gathers by the same way facts about e.g. the stability, energy consumption, and duration of a combined new component (pair j and k). All these additional attributes are stored in the knowledge-base as part of the information-base of the fitness-agent j. This knowledge can be transfered to the information-base of the component-agent, or the organiser has access to the information-base of the fitness-agent. With the aid of this knowledge each component-agent j (and fitness-agent j) can calculate its state S_j in relation to the connection partner k. Such a state does not include only W_{jk}, but also the kind of pairwise message exchange (statistical view) and the matter of activities to join together with component k.

In the next step the individual information Inf_j is calculated by mutual message exchanges e.g. between component-agents under the regard of the state j and the so long acquired knowledge W_{jk}. Already at this level we assume that the individual information of the two agents might be different, since e.g. the message transfer was disrupted or the separate sensor readings produce different results. In order to resolve this conflict additional cognitive methods must be employed. Typically this might concern the stability of a pairwise connection. The applied cognitive procedures have to handle the two possible situations: (a) the construction process is continued since the joint is stable and the information is wrong (e.g. inconsistent sensor data), or (b) the component is indeed not intact and the developmental process can not be continued. In the latter case a new connection partner must be detected, the failing component removed and replaced by a new intact component .

3.3.4.2 Levels of Developmental Phases in Creation of an Organisms

The two cognitive methods mentioned previously can be considered as part of a process that many individual pairs of components join pairwise together usually with different joint attributes (e.g. different work spaces). The result of these mutual activities is again the definition of the individual states S_j, the knowledge W_{jk} and the information Inf_j, where all descriptions are acquired by message exchanges. If the set of pairwise connected components is completed then the first developmental phase is finished.

Before we pass to the next construction level we would like to summarise the principle cognitive methods of our approach. There exist two parallel processes to acquire evolutionary information and fitness information. Each process stops if its

information is minimised. Afterward the two information minima are compared by the evolution-agent respectively fitness-agent and the adjustment between both minimal information values is performed by "negotiations" between these two agents. A third authority like a deciding-agent is thinkable but we exclude such agents in order not to be too sophisticated in our approach description. The process to acquire the minimum of information is a learning process (typically by try and fail). It ends with the storage of the local minima of the evolutionary information respectively the fitness information. These two types of information collection are considered as two different information gathering cycles that are horizontally connected in one level and vertically connected between different levels.

In the second developmental phase the aggregation of all combined components to a pre-shaped organism takes place. On this second level similar processes (message exchange, knowledge acquisition, state description) are performed as on the previous first level, but the main difference lies in the fact that the information is different, the basic methods are not replaced. In each restricted region there are organising agents and fitness evaluating agents that gather all available information of their common regions. These agents again arrange a distributed information storage by message exchange that delivers facts about the stability, etc. of the "erected" structure. Afterwards the selection of correct and incorrect operating parts and the replacement of the defect connections (if possible) or the repair of not intact parts (if possible) goes on by the pattern (a) or (b) mentioned before. The developmental process of the second level is repeated as long as the stability, adaptability, and survival of a pre-formed organism is guaranteed in a given environment.

On the third level the significant regions defining the global structure of the organism are fixed and the distinguished components that create the structure of the organism are determined. In addition the "interfaces" between the basic components are established. Further the basic parts of these components are determined. These are primarily the "differentiated cells" that later on build up all different components (including all sensors), but also elementary parts that can develop the "organs" of an organism. These are very important for the viability of the finished organism. This assembly process of the whole structure of the organism and its ability to operate together is again "managed" by the set of evolution-agents (organisers) and the set of fitness-agents.

Very typical operations of this phase are the operators for differentiation and reproduction. Those "cells" are differentiated that are in particular suited to generate the different components (for example wheels, body, arms, head, and eyes (sensors) of an artificial creature) and the organ-like internal components like power supply and communication links. The fitness-agents have to inspect whether all components fit together to sustain the whole organism, and the differentiated bonded components ("cells") are e.g. robust and durable enough to develop a whole, pre-structured organism. If the correct functionality and fitness under given environmental restriction is achieved then all "cells" are reproduced. An elementary model for the reproduction is a finite automaton or a Turing machine.

On the fourth level of the construction process occurs the inspection of the more global functionality of the whole organism together with the compatibility of

function and structure. A typical function is the ability to move in order to survive (e.g. search of foraging grounds). Here we need, in a more visible way, again the engagement of fitness-agents, since it might be possible that the constructed organism, especially the components that are foreseen for mobility, cannot perform this task. By a direct interaction between the two types of agents a rearrangement of the second level has to be activated. The result will usually be a new aggregation with e.g. passive and active joints. If this is possible then on the fourth level of this new evolutionary information and fitness information are stored. If a movable aggregation is not possible since e.g. only a tree structure is possible.

Typical operations of such a rearrangement are the operators for selection of the before differentiated elements and their reproduction. Only such combinations are selected that are able to work as hinges, stability elements, or gears for the movement, etc. The fitness-agents have to inspect whether the selected bonded components are e.g. robust and durable. If the correct functionality and fitness under given environmental restriction are achieved then these parts are reproduced.

If the new aggregation is considered in the light of exploration since the available components do not fulfill the demanded functionality then the operator of experimental crossover is engaged in order to generate new types of components that are tested again. If the tests are satisfied then again the selection and reproduction processes start. Here the minimisation of information again stops these processes.

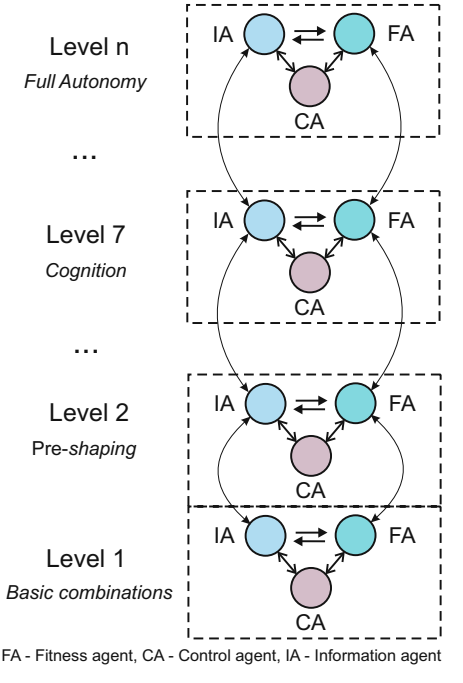


Fig. 3.25 Hierarchical, cascaded model of the development of an organism (thanks to Mr. Oliver Zweigle).

On the fifth level the coordinated regulation of all body components that are articulated occurs. The evolutionary information describes the closed loops that exist, and the fitness-information stores the parameter that optimise the loop. The interaction patterns between the evaluation-agents and the fitness-agents obey the same scheme as on all levels. The resulting total information of this level is the ability to control the orchestra of all articulated components of an organism.

On the sixth organism level the perception of all available sensor data, and, more important, the cognitive fusion of the received signals are performed. This is the first cognitive level of the organism in the sense of the implementation of an internal world model. It follows the seventh level where the internal world model is used to create different behavioural patterns (e.g. social behaviour). The eighth and all other succeeding levels implement more and more powerful cognitive abilities.

Fig. 3.25 summarises our statements as a cascaded model of two different types of evolution-cycles and fitness-cycles that are lateral and vertical connected. In addition there are control-loops that supervise the articulation of organism-components, of synchronised body movement, of "organs", of behaviour and so on. These control-loops are connected with the two other cycles in each layer since e.g. a structural change of an organism automatically tighten a revision of all three cycles. All lateral and vertical connections are driven in both directions, this means e.g. that a rearrangement of level 4 by level 2 is possible and foreseen. This model can also be implemented by layers of different neural nets that are in addition mostly recurrent.

Here we would like to remark that more cognitive agents can benefit of the pure mathematical algorithms of inanimated nature of Sect. 3.3.1. The processes that support the information collection will get more and more powerful and efficient if the cognitive abilities to handle the information e.g. in order to evaluate the context or making decisions grow up. One example of the optimising handling of information was given by the interactions of evolution-agents and fitness agents. The reader who prefers the more conventional view of information handling and artificial intelligence can consider it in direction of the more and more complex interactions between the two just mentioned types of agents in order to generate a total Turing test that is applied to machines (robots) and evaluated by machines and not humans.

3.3.4.3 Adaptability and Survival of an Artificial Organism

On the molecular level the connection of H_2O connection to water is a perfect example of adaptation to constraints of inanimated nature. But also in human bodies it plays a dominant role. In both cases it is adaptability to react on parameter changes. In this sub-chapter we investigate the adaptability of an organism on a higher level e.g. ability of structural changes (morphological ability). The ability for adaptation is closely connected with evolutionary processes and these are again strongly associated with the ability to survive e.g. in a rough ambiance.

3.3.4.4 Adaptability of an Artificial Organism

Here we try to describe a model of adaptability that is oriented on the MAS-model and the architecture of an organism that has been presented in the previous chapter. In our approach adaptability is the reorganisation of the acquired total information on each level or in dedicated levels in such a way that the organism is fit to respond in an appropriate manner if e.g. environmental alterations occur. This definition is in the sense of the formal definition of the behaviour scientist Conrad or it can be considered in terms of neuro-plasticity (change of brain structure to a better cope with external or internal alterations) or even in the meaning of phenotypic plasticity (determination of the phenotype by the genotype, e.g. Lamarckian learning). An adaptive organism can supplementary also be flexible in the range of the motion of a joint or it has a sheltering skin on the outside (in both examples temporarily modifications revert back to the original position).

In more technical expressions this means that we can define a kind of taxonomy in two directions. The first way to describe this taxonomy is to look from outside to inside. Here we consider the type of external restrictions and look for the adaptation operations of the organism in order to fulfil these new challenges. So a parametric change like a noticeable temperature increase can be coped with parameter variations without changing all three loops in the affected tiers. If for example a behavioural change is indicated or even a functional alteration is required then all three cycles on different levels must be changed. By application of our information-based view of adaptability this means that the number of all levels and cycles within all existing levels are fixed. The methods to acquire the new information are not changed, there are only new information values. For example the different internal states of the knowledge W_{jk} are not changed but their probabilities get modified values. In the same manner also the states S_j and finally the information Inf_j of an agent j will be modified in that way that the new total "fitness-information" of agent j gets very high (bad fitness) and the "evolution-information" is not dominantly changed. But by the interaction of both cycles the "fitness-information" is decreased (better fitness) by the start of a new evolutionary process. This interaction goes on in a recursive manner as long as the combined information (evolution and fitness) is as small as possible.

Now it might happen that the acquired information of an organism is not adaptive enough to match the new requirements. This is mainly true if extreme form changes are required (e.g. doubling of the height) or new functional conditions must be fulfilled (e.g. transfer from legs to wheels in order to increase the speed). In this case it might be that the principle architecture is powerful enough to perform the required alterations by the collection of new information in order to change all cycles. This corresponds to the generation of new regulative structures and rules.

In the last and most interesting case the whole or main part of the architecture is not able to cope with the completely new requirements, and therefore new methods for information gathering, new layers and within these layers new cycles and new horizontal and vertical connections must be evolved. The result of such an evolving and fitness evaluating process is creation of a new quality of information and as a

consequence of this process a new architecture of the organism. Such a phenomena is called emergency and this is the modality how natural beings are created.

This emergency is unfortunately till now in a big scope like a miracle. Please just think how a set of thousand of eye cells coordinate themselves in order to be an eye. Even if an eye is created in the head then a complete new information collection and information handling must evolve in order perform the interpretation of the images. This is one "hour of birth" for the creation of the cognitive capability of perception that can only be done by one or more layers in the architecture of an organism. Here we hope that with our MAS-model of the previous chapter it will be incrementally possible to bring light in such an evolutionary process.

The taxonomy of adaptability will be completed if we describe this ability in a reverse way that points from inside to outside. For sake of brevity we skip this description here.

3.3.4.5 Survival of an Artificial Organism

The adaptability of an organism is closely connected with the homeostasis of the organism that is given by all inner "organs". Only if the homeostasis is able to generate and maintain stable states then an organism can survive. For example a robot will "die" if it gets not enough power supply (or e.g. a man gets no more air to breathe). The homeostasis of an organism is primarily defined by the control cycles for energy harvesting from outside, for energy distribution in the organism, for message passing in the body, etc. In mathematical terms these mechanisms for adjustments and regulation can be described by the methods of nonlinear dynamics. So we are searching for the invariant set of fixed points (set of dynamic equilibria), for bifurcation diagrams, the presentation of extended phase diagrams and for the behaviour of the solution if parameters are changing(for more details see Sect. 1.2).

If we try to go a similar way to define a taxonomy for survival changes then we have at first to consider the adaptability of the "inner organs" to the parameter changes caused by environmental alterations and task that an organism has to perform under the given conditions. The organism cannot survey if the given constraints are too extreme and can therefore not be fulfilled if all other activities to collect and handle information in the different layers to regulate the control-loops go on unaltered. If the restrictions are not so severe that an organism "dies" and our last assumption is still true then dysfunction in the movement, in the behaviour, and in the functionality of the organism occur.

Then the next step would be to change all regulatory controls but not change the whole architecture of an organism. This step is similar to that one we have done in the interrelationship to adaptability. This might be comparable with a strong training of a sportsman. In technical sense the frequency of the cycles can be increased, the power consumption can be enlarged by new accumulators, the frame rate of a camera might be reduced, and finally the body increases the pressure on the joints.

The appearance of new "organs" can also be stipulated by minimal survival changes (the organism must survive) and there is no possibility that an organism can adapt to the new challenges. Then an evolutionary process together with a

fitness process must be activated to acquire the new information and create new differentiated "cells" that can replace the old organ at the same position in the organism. If this is not possible the whole structure of the organism must be redesigned by the evolution in order to find a new site for the new "organ". A similar process has to be launched if e.g. components of the organism are missing in order to survive. A robot that cannot see, grip and move cannot survive if the power sources are not reachable to it. It must then change its form meaning that new components have to be developed and then attached to a new body. And again the interaction between evolution and fitness must evaluate the new "products" as long as the robot can see, grip and move somehow. If this is achieved then the optimisation goes on recursively until a (may be local) optimum is found. The result of such optimisation process delivers very different components depending from the environment.

In animated nature we have perfect examples for such emergency processes. For fishes the eyes (e.g. in deep see), arms (flippers) and legs (tail fin) are different from that one of birds (sharp seeing eyes, wings and legs), and these structuring elements are again modified e.g. for horses that live on the ground(small viewing angles, no arms but four legs). There strong efforts in our laboratory to use this biological based inspiration to rebuild artificial creatures as bees, fishes, ants, and robots in order to study how these evolutionary "models of success" can be constructed and prosperous applied in different applications of all three mentioned different environments.

Chapter 4
Adaptive Control Mechanisms

4.1 General Controller Framework

Guy Baele, Yao Yao, Yves Van de Peer, Alan Winfield, Serge Kernbach

Various control mechanisms can be used to control robotic entities. In this chapter, we consider two different classes of such control mechanisms: *bio-inspired* such as hormone-based controllers in Sect. 4.2, embryogenetic controllers based on ANN in Sect. 4.3 and artificial immune systems in Sect. 4.4; and *tech-inspired* such as morphogenetic controllers in Sect. 4.5 and controllers for adaptive macroscopic locomotion in Sect. 4.6. The split between bio-inspired and tech-inspired strategies is intentionally chosen for several reasons: in many situations both strategies supplement each other, they correspond to differences between SYMBRION and REPLICATOR concepts, moreover this allows performance comparison of different strategies depending on environmental conditions. However, independently of the strategy used, these controllers can use different evolutionary (Baele *et al.*, 2009) and self-organizing mechanisms (Kernbach *et al.*, 2009b).

In this section we consider a general controller framework: Sect. 4.1.1 describes problems, challenges and design principles of such a framework, Sect. 4.1.2 introduces the biological inspiration for the artificial genome, and Sect. 4.1.3 introduces the action-selection mechanism. Finally, Sect. 4.1.4 provides an overview of the above mentioned controllers.

4.1.1 Controller Framework in SYMBRION/REPLICATOR

In robotics, several different control architectures are well-known, as e.g. subsumption/reactive architectures (Brooks, 1986), insect-based schemes (Chiel *et al.*, 1992) or structural, synchronous/asynchronous schemes, e.g. (Simmons, 1991). An overview of these and other architectures can be found in (Siciliano & Khatib, 2008). Recently, multiple bio-inspired and swarm-optimized control architectures have appeared, e.g. (Tianyun *et al.*, 2008), (Kernbach *et al.*, 2009c). In designing the

P. Levi and S. Kernbach (Eds.): Symbiotic Multi-Robot Organisms, COSMOS 7, pp. 229–336.
springerlink.com

general control architecture for the SYMBRION/REPLICATOR system, we face several essential challenges:

Multiple processes	Artificial organisms execute many different processes, such as evolutionary development, homeostasis and self - organizing control, learning, middle- and low-level management of software and hardware structures. Several of these processes require simultaneous access to hardware or should be executed under real-time conditions.
Distributed execution	Hardware provides several low-power and high-power microcontrollers and microprocessors in one robot module. Moreover, all modules communicate through a high-speed bus. Thus, the multiprocessor distributed system of an artificial organism provides essential computational resources, however their synchronization and management present a challenge.
Multiple fitness	Fitness evaluation by using local sensors is already mentioned in Sect. 2.2.3. Here we need to mention the problem of credit assignment related to the identification of a responsible controller, see e.g. (Whitacre *et al.*, 2006). Since many different controllers are simultaneously running onboard, the problem of credit assignment as well as *interference between controllers* is vital.
Hardware protection	Since several controllers use the trial-and-error principle, the hardware of robot platform should be protected from possible damage caused during the controllers' evolution.

An overview of the hardware architecture and comparison to other reconfigurable robot systems is given in Sects. 2.2.1 and 2.2.2. Corresponding to the hardware architecture, the general controller framework is shown in Fig. 4.1. This structure follows the design principles, originating from *hybrid deliberative/reactive systems*, see e.g. (Arkin & Mackenzie, 1994). It includes a strongly rule-based control component, see e.g. (Li *et al.*, 2006) as well as multiple adaptive components (Kernbach *et al.*, 2009b). The advantage of the hybrid architecture for SYMBRION/REPLICATOR is that it combines evolvability of reactive controllers, and their high adaptive potential, with deliberative controllers that provide planning and reasoning approaches required for the complex activities of an artificial organism.

Controllers are started as independent computational processes, which can communicate with each other and with different sensor-fusion mechanisms, such as virtual sensors, see Sect. 3.2 or the world model, see Sect. 3.1. Processes are running on different modules, synchronization and interaction between them is performed through message-based middleware system. There are controllers, which use evolutionary engines, see more in chapter 5, and their structure is coded in the artificial genome. Several bio-inspired ideas towards such an artificial genome are described in Sect. 4.1.2. It is assumed that there are also a few task-specific

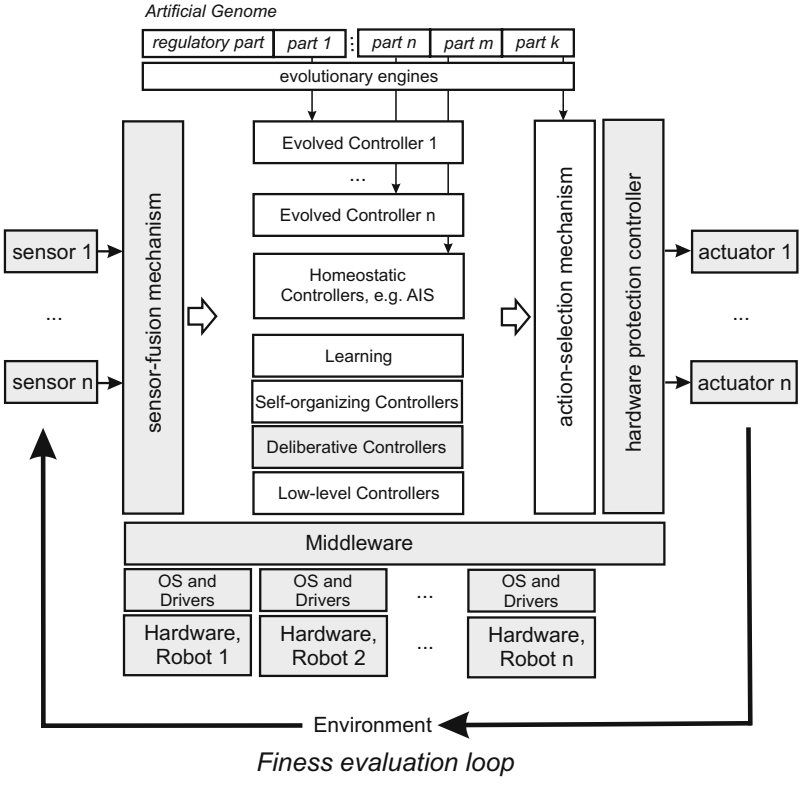


Fig. 4.1 General controller framework in SYMBRION/REPLICATOR. All controllers/processes are distributed in the computational system of an artificial organism, OS – operating system. Structure of controllers utilizes hybrid deliberative/reactive principle.

controllers, which are placed hierarchically higher than other controllers. These task-specific controllers are in charge of the macroscopic control of an artificial organism. They may use deliberative architectures with different planning approaches, see e.g. (Weiss, 1999).

The action-selection mechanism is one of the most complex elements of the general controller framework. This mechanism reflects a common problem of intelligent systems, i.e. "what to do next", see (Bratman, 1987). An overview of the action-selection mechanism is presented in Sect. 4.1.3. Finally, a hardware protection controller closes the fitness evaluation loop for the evolvable part of controllers (Kernbach *et al.*, 2009a). This controller has a reactive character and monitors activities between the action-selection mechanism and actuators as well as exceptional events from the middleware. It prevents actions that might immediately lead to destroying the platform, e.g. by mechanical collisions.

4.1.2 Bio-inspiration for the Structure of Artificial Genome

As shown in Fig. 4.1, an artificial genome stores the information for the evolvable controllers. In (molecular) biology, the genome is the entirety of an organism's hereditary information, encoded in DNA (or RNA). The genome includes both genes (coding regions) and non-coding regions in its DNA. The information encoded in the genes is used in the synthesis of a functional gene product by a process called "gene expression". Such gene products are often proteins (or functional RNAs for non-protein coding genes), essential parts of organisms which participate in virtually every process within cells. Cells are the basic structural and functional units of all known living organisms and are often called the building blocks of life.

Within cells, deoxyribonucleic acid (DNA) is organized into structures called chromosomes. DNA is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms and some viruses. The main role of DNA molecules is the long-term storage of information and contains the instructions needed to construct other components of cells, such as proteins and RNA molecules. The information in DNA consists of a sequence of four bases (A, C, G and T).

In evolutionary robotics, different representations can be used to store a robot's artificial genome (illustrated in Fig. 4.2). Common to these approaches is the presence of a number of genes in the artificial genome, which possibly also consists of a number of chromosomes. The most biologically plausible approach is to use a string of digits (i.e. 0, 1, 2 and 3 instead of A, C, G and T) to encode the information. Such an approach requires biological structures, such as promoter regions or so-called start codons, to be present in this string of digits to indicate where each gene begins and ends (an artificial genome based on such principles was used in the work of (Reil, 1999) to simulate a gene regulatory network). This way, the evolutionary engine can parse the information present in the genome when constructing a robot's controller. Such a bio-inspired approach to an artificial genome allows for the design of mutation and variation operators that are known to have occurred through time in biological evolution.

A bit string can also be used to store information in an artificial genome. Given that unique patterns of bits cannot be encoded this way, for example to indicate

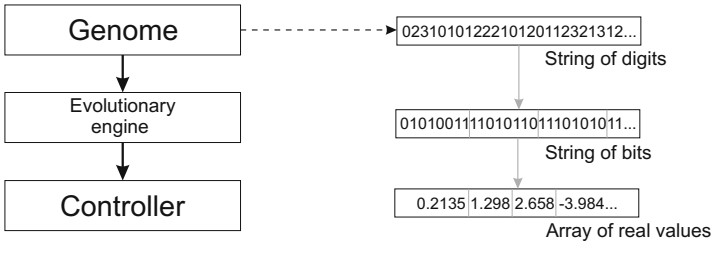


Fig. 4.2 Different representations of an artificial genome that need to be transferred, by the evolutionary engine, to the controller.

the start position of a gene, fixed length genes (and hence fixed length genomes) are often used (see Fig. 4.2). Such a binary representation is easily translated to real values which can then be used in a robot's controller, for example as synaptic weights in an artificial neural network (ANN; see Sect. 4.1.4). Mutation and variation operators for this type of representation then consist of changing the state of one or more bits in the sequence (Calabretta *et al.*, 1996; Calabretta *et al.*, 1997; Calabretta *et al.*, 2000; Calabretta *et al.*, 1998). To avoid the translation process into real values, often the artificial genome contains these values (for example, the synaptic weights for a neural network) directly in an array, see e.g. (Bredeche *et al.*, 2009).

Of these three types of encoding, the string of digits is the approach most able to mimic biological reality and incorporate genomic properties inherently in the representation. Instead of straight forwardly encoding real values, as the bit-string representation does, more complex systematics for constructing a robot's controller can also be incorporated in the sequence of digits. For example, in biology, gene regulation is essential for viruses, prokaryotes and eukaryotes as it increases the versatility and adaptability of an organism by allowing the cell to express proteins when needed. Furthermore, gene regulation drives the processes of cellular differentiation and morphogenesis (the biological process that causes an organism to develop its shape), leading to the creation of different cell types in multi cellular organisms where the different types of cells may possess different gene expression profiles even though they all possess the same genome sequence.

Such a genetic regulatory system can be modelled in an artificial genome, see (Reil, 1999), adding different elements to the genome such as promoter regions, binding sites and the gene products that are the result of a gene regulatory network (GRN). Indeed, Reil has developed an artificial genome with biologically plausible properties to study gene expression(Reil, 1999). The author uses the concept of a standard promoter to define genes in the genome and achieves gene regulation by the binding of gene product to matching sequences of the artificial genome, corresponding to the action of transcription factors in DNA-based genomes. More recently, other approaches have been developed to study the evolution of genetic regulatory network (see e.g. (Kuo *et al.*, 2004; Quayle & Bullock, 2006)).

A GRN approach can be also be used to regulate a collection of genomes (or parts of genomes). Indeed, as can be seen in Fig. 4.1 different genomes can be used, resulting in different controllers for a given robot. This collection of genomes may be prefixed by a regulatory part that regulates the expression of each artificial genome. In its simplest form, the regulatory part provides the necessary information to decide which genome (or which part of the genome) will be expressed as a controller (before the action-selection mechanism takes place, see Sect. 4.1.3). More complex regulatory mechanisms at this level are also possible. For example, the regulatory part could indicate that parts of a given genome (i.e. one or more of its genes) should be more expressed, resulting in a different robot controller.

A biologically plausible artificial genome allows us to incorporate many interesting features for a given robot controller. Further, such a representation allows for a straightforward implementation of bio-inspired operators. For example, the process of gene duplication (i.e. any duplication of a region of DNA that contains

a gene) has been shown to play a major role in evolution and is easily translated onto a bio-inspired artificial genome. However, a common (biologically plausible) representation for a series of different robot controllers as suggested in Fig. 4.1 is far from trivial. Different controllers require genomes that may need very specific mutation and variation operators, so even though the genome representation may be highly similar, the operators may not be. Well-known control mechanisms such as artificial neural networks (ANNs) may not even benefit from such a bio-inspired genome representation since adequate convergence towards optimal fitness values can be reached without such a representation (see e.g. (Floreano *et al.*, 2008a)).

4.1.3 Action Selection Mechanism

Formally, action selection is defined as follows: "given an agent with a repertoire of available actions ... the task is to decide what action (or action sequence) to perform in order for that agent to best achieve its goals" (Prescott, 2008). Within the context of the SYMBRION/REPLICATOR general controller framework shown in Fig. 4.1, the role of the action selection mechanism is to determine which controller(s) are driving the actuators at any given time. At one level the action selection mechanism can be thought of as a switch, selecting which of the controllers is connected to the actuators; however a simple switch would fail to provide for, firstly, smooth motor transitions from one controller to another and, secondly, the fact that in this hybrid deliberative/reactive architecture some controllers will need to be prioritised for short time periods (e.g. for obstacle avoidance) whereas others need periods of control over longer time spans (perhaps subsuming low-level reactive elements) to achieve high level goals. In practice, therefore, the action selection mechanism will need to combine some or all of the following elements:

- prioritisation of low-level reactive controllers so that they are given control with very low latency;
- vector summation or smoothing between some controller outputs in order to achieve jerk free motor transitions on controller switching, and
- a time multiplexing scheme to ensure that different controllers are granted control with a frequency and for time periods appropriate to achieving their goals.

Action selection mechanisms have been the subject of research in both artificial and natural systems for some years, see for instance (Maes, 1990; Hexmoor *et al.*, 1997; Prescott *et al.*, 2007). However, in a recent review Bryson suggests that no widely accepted general-purpose architecture for action selection yet exists (Bryson, 2007). Relevant to the present work is a review of compromise strategies for action selection (Crabbe, 2007). A compromise strategy is one in which instead of selecting a single controller, the action selection mechanism combines several controller outputs in such a way as to achieve a compromise between their (otherwise conflicting) goals; (Crabbe, 2007) suggests that a compromise strategy is more beneficial for high-level than low-level goals.

It is important to note that the action selection mechanism embeds and encodes design rules which will critically influence the overall behaviour of the robot. In

order to arbitrate between, possibly conflicting, controller goals the action selection mechanism will certainly need to access internal state data for the robot (i.e. from the homeostatic controllers), and may need to access external sensor data. Furthermore, given that those action selection design rules and their parameters may be difficult to determine at design time, we are likely to require an evolutionary approach; hence the connection between the genome structure/evolutionary engine and the action selection mechanism shown in Fig. 4.1. We may, for instance, evolve the weights which determine the relative priority of controllers as in (González *et al.*, 2006), or co-evolve both controllers and action selection parameters (González, 2007).

4.1.4 Overview of Different Control Mechanisms

This section briefly overviews several controllers, discussed further in this chapter in more detail.

ANN-based Controllers

In many cases, the control systems of evolutionary robots are artificial neural networks (ANNs), which are used in many controllers. An ANN is a collection of units (neurons) connected by weighted links (called synapses or connection weights) used to transmit signals. Input and output units receive and broadcast signals to the environment, respectively. Oftentimes, the input signals correspond to reads from proximity (or distance) sensors and the output signals correspond to commands for the robot's actuators (or wheels). Input sensors can also include cameras or light sensors, actuators can include screw drives or docking elements. The output of a unit can be the sum of all incoming signals weighted by connection strengths. The behavior of a neural network is largely determined by the values of connection weights, also known as synaptic strengths, and except for a few simple cases, it is difficult to determine the strengths required for achieving a desired input/output mapping (Nolfi & Floreano, 2000a). The number of connections that need to be optimized is often relatively small. On many occasions, a fully interconnected network is used (connecting each input neuron to each output neuron, with the addition of a bias neuron for each output neuron). More complex network topologies with, for example, additional hidden layers of neurons are however possible. For many cases, simple neural networks have been shown to work well, as we will illustrate in this section. Different robots are equipped with different input and output possibilities. As an example, we have performed a simulation experiment with E-puck robots. Fig. 4.3 shows a schematic representation of the eight infra-red (IR) proximity sensors of the E-puck robot (Mondada *et al.*, 2009) and the artificial neural network connecting these eight input units to the robot's two wheels (or output units).

Given that the initial population of robots for our example (as for most simulation experiments) consists of randomly generated artificial neural networks, an evolutionary strategy is required to obtain a population of well-performing robots (measured using a fitness function) after a given number of generations. Such a strategy proposes changes to the connection weights of the network, but might propose

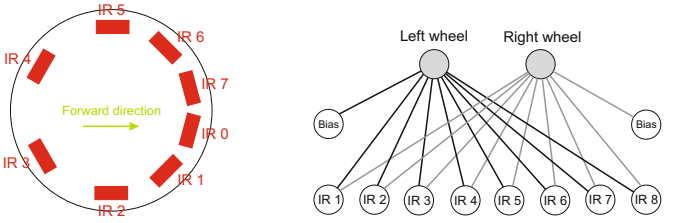


Fig. 4.3 Left: schematic display of the proximity sensors on an E-puck robot. Right: artificial neural network (ANN) as the controller of choice for the E-puck robot.

topological changes as well. It can however take a large number of generations before the population evolves the required properties. Determining a well-performing evolutionary scheme may not be feasible on real robots, as the serial evaluation of individuals through a large number of generations might require a long time when only a limited number of physical robots are available. One way to avoid this problem is to evolve robots in simulation and then run the most successful individuals on the physical robot. This way, the power of parallel computers can be exploited to run more individuals at the same time.

In our example of ANN-based controllers, we consider experiments with 50 robots in simulation. The connection weights are evolved during 100 generations by using a mutation operator (no cross-over). We have performed the simulations in the Player/Stage simulation environment (Gerkey *et al.*, 2003) and the neural network shown in Fig. 4.3 is used. We assume real values for these connection weights and have implemented the mutation operator as a Gaussian mutation, meaning that a value drawn from a Normal distribution $N(0,\sigma)$ is added to the connection weights undergoing the mutation. Small values for σ will only allow small changes to the connection weights to be tested. This can prevent the robot population from reaching optimal fitness levels but also prevents disadvantageous mutations from occurring (which decrease the robots' performance). Large values for σ will result in more disadvantageous mutations but will also prevent the robot population from getting stuck in local optima in terms of fitness level. Each connection weight is allowed to evolve with a given probability (which we assume to be equal for all weights).

Fig. 4.4 shows peak and average fitness values of a simulation experiment in a square area with two (rectangular) obstacles. The objective for each robot is to avoid the obstacles in the area and explore the area without getting stuck. The robots are also expected to cover as much territory in the simulation area as possible. Two simulation experiments that last 100 generations for a population of 50 robots are shown in Fig. 4.4, for a mutation operator with $\sigma = 0.1$ (left part of the figure) or $\sigma = 0.5$ (right part of the figure) and a fixed mutation probability of 10%. The first experiment, with $\sigma = 0.1$, has problems reaching optimal fitness levels because the proposed values for the connection weights do not differ much from the current values. The second experiment, with $\sigma = 0.5$, overcomes this problem by allowing more drastic mutations to occur. As can be seen in generation 55, large mutations may have a drastically negative influence on both peak and average fitness.

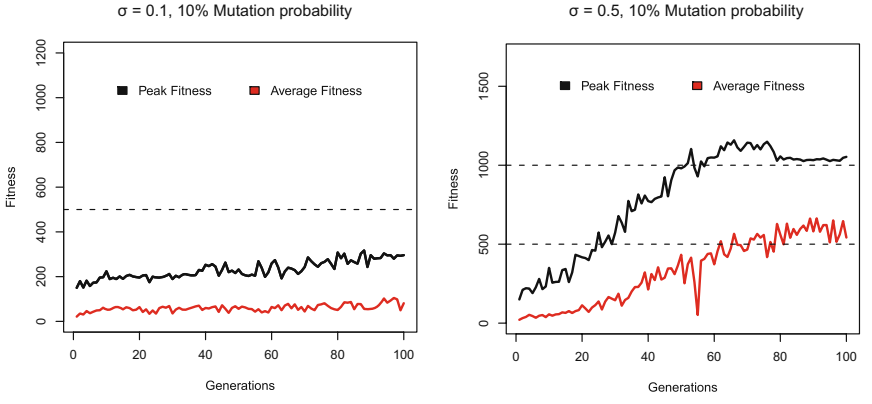


Fig. 4.4 Peak and average fitness curves for an obstacle avoidance / area reconnaissance simulation experiment where a connection weight of the ANN has a probability of 10% to be updated with $\sigma = 0.1$ (left) and $\sigma = 0.5$ (right).

Hormone-based Controller

Homeostasis is the ability of an organism to achieve a steady state of internal body function in a varying environment. The endocrine system and the immune system are central to an organism's ability to maintain homeostasis. Within an organism hormones implement a regulatory mechanism acting directly at the level of individual cells. Hormones are chemicals, released by one or more cells, that affect cells in other parts of the organism. The endocrine system is responsible for the production and storage of these hormones. Cells respond to a hormone when they express a specific receptor for that hormone. Hormones have many functions which affect behaviour, stimulate or inhibit growth or control the reproductive cycle. Hormones are released into the blood or lymph system and are able to reach virtually all the tissues within the organism. The response to hormones is highly specific, i.e. not all cells react to all hormones (Neal & Timmis, 2003).

Biosynthesis, a phenomenon wherein chemical compounds are produced from simpler reagents, is responsible for the production of hormones. Sensor inputs trigger secretions of hormones which are then transported across the organism. When a hormone locates its particular target cells, binding takes place through specific receptors on the cells. When a hormone binds with a receptor on the cell membrane, it stimulates internal signals to the appropriate sites within the cell, which in turn alter the cell's activity. Hormones decay over time and are ultimately removed from the organism.

Fig. 4.5 shows a reproduction of a representation of the hormone-based robot controller introduced by Schmickl and Crailsheim (Schmickl & Crailsheim, 2009), where hormone input is triggered by sensor inputs. Sensors only affect the hormone secretion in the compartment they are associated with, but all hormones spread through the system by a diffusion process. Fig. 4.5 shows three virtual compartments, of which the central compartment acts as a time-delaying compartment for

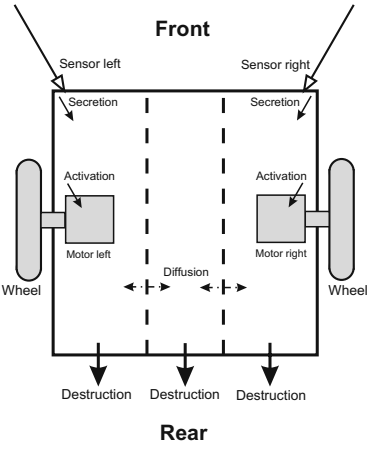


Fig. 4.5 Schematic representation of a hormone controller, assuming an internal compartmentalization of the robot's body. Sensors excrete hormones into one compartment, actuators get activated by local hormone concentrations and hormones diffuse between compartments.

the diffusion processes. A full description of an Artificial Homeostatic Hormone System (AHHS) is discussed in Sect. 4.2.

Embryology-based Controllers

As discussed, the morphological structure of an ANN is very important for its functionality as basic structural features of such networks can determine the basic capabilities of the network (Elman, 1990). Frequently used approaches include finding optimal values for the connections of fully-linked networks of cells using a genetic algorithm and manually defining the network structure. The former approach has the disadvantage that increasing numbers of cells lead to a drastic increase in computation time while the latter approach has a poor ability to adapt to unknown situations or problems that were not accounted for when designing the network.

Sect. 4.3 discusses a method to organize the nodes and links of an ANN using virtual embryogenesis. Embryogenesis is the process by which an embryo is formed and develops. The virtual embryo consists of individual cells, which can develop to nodes in the ANN during the embryologic process. These cells can duplicate, die, specialize, emit chemical substances or build links to other cells. It is these links that represent the connections between the nodes of the resulting ANN. An important aspect of the approach discussed in Sect. 4.3 is the abstraction of complex biological processes because of the need for fast simulations and the limited hardware resources available in robotic systems.

Immune-system-based Controller

Like the endocrine system, the immune system plays an important role in an organism's ability to maintain homeostasis. The immune system is an organism's defense against attacks by "foreign" invaders (called pathogens). Through a series of steps called the immune response, the immune system attacks organisms and substances

that invade a system and cause disease. The immune system is made up of a network of cells, tissues, and organs that work together to protect the body. The immune system detects a wide variety of agents, from viruses to parasitic worms, and needs to distinguish them from the organism's own healthy cells and tissues in order to function properly.

The natural immune system protects organisms from infection with layered defenses of increasing specificity. Most simply, physical barriers prevent pathogens such as bacteria and viruses from entering the organism. If these barriers are breached, the innate immune system, which is found in all plants and animals, provides an immediate but non-specific response. However, if the innate response is evaded, vertebrates possess a third layer of protection (the adaptive immune system), which is activated by the innate response. The adaptive immune system "adapts" its response during an infection to improve its recognition of the pathogen. This improved response is then retained after the pathogen has been eliminated, in the form of an immunological memory, and allows the adaptive immune system to mount faster and stronger attacks each time this pathogen is encountered.

Disorders in the immune system can result in disease. Immunodeficiency occurs when the immune system is less active than normal, resulting in recurring and life-threatening infections. Immunodeficiency can be the result of a genetic disease. In contrast, autoimmune diseases result from a hyperactive immune system attacking normal tissues as if they were foreign organisms.

A robotic swarm can be defined as a society of robots that coordinates its behaviour via interaction and cooperation with other robots in the society to reach a common goal. The robots are constantly exposed to changes in their environment, which can alter the states of the robots. An artificial immune system is therefore necessary to assure that those changes will not affect the robots' behaviour in a negative way, both in organism and in swarm mode, so that a given goal can still be accomplished. A detailed description of AIS is discussed in Sect. 4.4.

Morphogenetic and Locomotive Controllers

The robot organism can function in either swarm mode or organism mode. When a situation is encountered where one or more single robots fail to reach a given goal, the robots can enter "organism mode" in order to form a morphology that will allow the collection of robots to reach the goal. The search space to evolve such morphologies is not large, especially when the symmetry of modules is taken into account. Three different subsets of all possible topologies are discussed in Sect. 4.5, i.e. pattern-based, forbidden/non-efficient and specific topologies. The pattern-based subset is the largest one, containing variations of *a priori* known efficient patterns. These pre-generated patterns lead to a range of new topologies by performing small perturbations. In Sect. 4.5, it is shown that a set of pre-evolved solutions (along with topologies that are deviations from these solutions) will be beneficial in terms of resource requirements. An important consideration for morphogenesis is the structural

stability of organisms. In order to provide stability several element in the structure should be strongly connected by, for example, making multiple connections between them.

From a mechanical point of view such multi-robot organisms consist of connected joints and links, which are described by kinematic parameters. However, this description is troublesome as the morphology of the system is not known in advance. The number of possible configurations of the robots in organism mode requires adaptive model mechanisms to analyse such multi-body organisms, as traditional approaches will result in a high level of complexity. As discussed in Sect. 4.6, promising approaches stem from the theory of Lie groups and Lie algebras in combination with screw theory for body kinematics. Further, the restriction of most motion planning and motion control approaches to fixed kinematics requires an adaptive motion trajectory planning system for the multi-robot organisms.

4.2 Hormone-Based Control for Multi-modular Robotics

Thomas Schmickl, Heiko Hamann, Jürgen Stradner, Karl Crailsheim

In this research we aim for an application of multi-modular reconfigurable robots in a way that has not been targeted before. Our objective is to create a swarm of autonomous robot modules which dynamically aggregate to organisms and which also dynamically reconfigure themselves to other body shapes. In our approach, it will be possible to break one robot organism consisting of a given number of linked modules into two pieces. What makes our approach unique is the fact that our system is able to react in three different ways to such a disturbance:

- These two pieces will not only continue their "old" behaviours. Instead, both – now separated – parts will act like new organisms, change the mode of operation of their modules in a way that allows them to efficiently navigate and move in the environment.
- In addition, they are able to attract other singular, moving swarm modules to incorporate them to their body. Thus, they possibly grow back to their original size and form. This way we will be able to replicate one organism into two.
- Alternatively, the smaller separated organisms could locomote their bodies in a way that they approach each other and re-join to their original organism.

To achieve such behaviours in modular robotics, a high degree of decentralisation in the control process is needed. Controlling algorithms of autonomous modules should not be controlled by centralised decision making systems. Their behavioural programs of individual modules should be chosen according to the collective goal of the organism and according to their positions in the organism's topology. Thus, positional information should be used in communication instead of fixed ID numbers of modules.

In addition to these requirements, our target system should work on different levels. In singular moving robots, in swarms of singular robots, and in joined modules

that form an organism. For our intended reconfigurable modular robotic systems, we describe the "life cycle" of the robot swarm by the following phases:

Swarm phase: The modules have to move independently. They have to perform basic functionality, that is present in almost every swarm robotic system, e.g., collision avoidance and obstacle detection. To form a multi-modular robot, the independent modules have to aggregate, which means that they have to approach each other. If the body formation process should act in a "swarm-intelligent" manner, directed motion (taxis) is necessary as well as several forms of interaction and communication.

Body formation: The modules should arrange in the desired manner and, by linking to already aggregated modules, they should contribute to the growth of the multi-modular "organism". This process will again involve interaction and communication.

Organism phase: While the modules are aggregated, they have to move and behave in a coordinated manner to allow the desired collective movement of the organism. Again, this coordination requires interaction and communication among the organism's modules.

We continue with a detailed biological motivation of our approach and subsequently define our hormone-based control system: We describe first the sources of inspiration that led to the development of the AHHS controller. In the next section, we discuss the existing state-of-the-art in the field of bio-inspired robot control and in multi-modular reconfigurable robotics. Then we describe the mathematical formulation of the robot controller, as well as the definition of the data-structure that we call "genome". After this section, we give two examples on bio-inspired dynamic compartmentalization of the virtual inner space of the robot modules. Then, we describe our evolutionary mechanism that produces behavioural programs in the AHHS. We describe several examples of such evolved behaviours for singular robots and for joined robot organisms. In addition, we show the major feedback loops that govern the overall system and discuss implications and outlooks of our AHHS technology.

4.2.1 Micro-organisms' Cell Signals and Hormones as Source of Inspiration

Although today's robotic modules are more sophisticated than in recent decades, they are still *far behind* the capabilities of real organisms. Thus, in our case of bio-inspired systems, it is a good idea to search for organisms that are rather old from an evolutionary perspective and simple from a morphological and physiological perspective.

Unicellular life forms like algae, bacteria, protozoa and slime mould have evolved billions of years ago, and they are able to show all the required processes mentioned above: movement of singular cells, taxis (movements directed by environmental stimuli), aggregation of organisms, coordination among cells, communication among

cells and even joining to organism-like "colonies" which are able to perform taxis and aggregation also at the colony level.

To achieve these features, they do not exploit neurons, brains or muscles. In contrast, it is a set of surface receptors, cell signalling (cell hormones), and simple actuators (flagella, cilia) that allow all these collective behaviours.

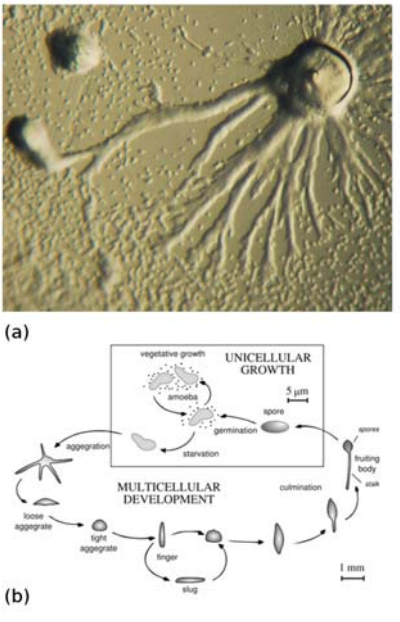


Fig. 4.6 (a) Aggregation of slime mould amoebas. A chemical signal (cAMP) is excreted by amoebas and used as environmental gradient in positive chemo-taxis. (b) Life-cycle of the slime mould *Dictyostelium discoideum*. Source of both pictures: Wikimedia Commons.

The amoebas of the slime mould *Dictyostelium discoideum* and *Dictyostelium mucoroides* show fascinating collective abilities in their aggregation behaviour (Camazine *et al.*, 2003). They excrete a chemical substance, cyclic adenosine 3',5'-monophosphate (cAMP), at small portions periodically. If the cAMP concentration in the local environment exceeds a given threshold Θ, it excretes a much higher amount of cAMP molecules. Thus, it produces a chemical "pulse" signal in response to a received input signal. After such a pulse was emitted, an amoeba cannot respond with another pulse for a period of time, as it has to refill its cAMP depot by taking cAMP molecules from the local environment. After this refractory period, amoebas are ready to fire the next cAMP pulse, as long as the environment still holds a cAMP concentration above the threshold Θ. In a habitat with densely aggregated amoebas, cAMP concentration is high enough to repeatedly trigger this behaviour, thus one could interpret slime mould amoebas as acting like oscillators. These oscillators interfere with each other. Thus, they represent a network of coupled oscillators, which is able to produce chemical signals in a spiral-wave like pattern (Müller *et al.*, 1998).

In contrast, in habitats with low population densities, amoebas excrete chemical "pulses" not frequently, because cAMP concentrations do seldom exceed Θ. Thus, no spiral waves emerge, due to the low degree of connectivity between amoebas. In addition, non-refractory amoebas tend to move uphill (taxis) in the cAMP gradient and refractory amoebas tend to remain on place. This leads to the following group-level behaviours. In habitats with low amoeba density, amoebas walk uphill in local cAMP gradients and, thus, increase the density of amoebas (positive feedback). The steady degradation of local cAMP produces negative feedback, thus creating an equilibrium cAMP concentration that correlates with amoeba density.

Whenever enough amoebas have joined such a local aggregate, local cAMP concentration jumps quickly to a much higher level, thus attracting even more amoebas from other nearby aggregates. Due to the motion principles of amoebas, they move uphill in a trail-like formation towards the centre of aggregation, see Fig. 4.6(a).

At this location a three-dimensional snail-like pseudo organism is formed. This aggregation of up to 10^5 cells is usually called "slug" state of the slime mould. It is not a real organism from the biological point of view. In fact, it is an aggregation of cooperating cells, similar to our desired multi-modular robot organism. At this point of the development, the cells that form this slug split up into three groups independently from their (random) positions. They become "pre-stalk" cells, "pre-spore" or "anterior-like" cells (Siegert & Weijer, 1992). This differentiation process is correlated to chemo-tactic cell sorting, resembling a sort of low-level ad-hoc morphogenesis in the freshly formed pseudo-organism. The differentiated cells are located in different positions and are expressing different behaviours, just like we want our robot modules to reflect their position in the organism and to perform different behaviours at different locations.

Finally, the amoebas coordinate their movements and synchronise, so that the slug is able to move like one single organism. It is driven forward by spiral screw-like pattern of contraction (synchronised by cAMP secretion) which is still produced by the same simple amoeba-to-amoeba interaction rules. The only difference is that in the dense three-dimensional pattern of the slug phase, the initially flat spiral waves are automatically converted into a three-dimensional screw-like pattern (Dormann *et al.*, 1997; Steinbock *et al.*, 1993; Siegert & Weijer, 1992). This motion pattern allows these organisms to morph into a spore-releasing plant-like structure, which is important to close the life-cycle of this species, see Fig. 4.6(b).

Unicellular algae and bacteria also show very impressive capabilities concerning taxis (Zonia & Bray, 2009; Darnton *et al.*, 2007; Khan *et al.*, 1998). The bacterium *Echerichia coli*, see Fig. 4.7(a), is propelled by a set of flagella which can be actuated in two modes of operation: Counter-clockwise (CCW) and clockwise (CW). The first mode drives the bacterium almost straight forward, the second mode produces a sort of chaotic tumbling behaviour, because the flagella cannot synchronise well if they – or some of them – are rotated CW. This is due to the three-dimensional form of these flagella, thus these features basically derive from their chemical properties and from the process that constructs them. The bacterium moves uphill in the gradient of nutrients, which surrounds it in the local environment, by the following strategy. On the surface of the bacterium a set of "methyl accepting chemo-taxis proteins" (MCP) senses the environmental nutrient concentration, pH value and oxygen saturation. MCP are fixed on the outside of the bacterium cell, but reach through the membrane to the inner cytoplasm. Binding of substances to the outer parts of the MCP triggers chemical reaction on their inner parts, producing a chemical signal. These receptors induce a chain (network) of intra-cellular chemical reactions. This way all different signals originating from many MCP get integrated by diffusion in the inner space of the bacterium, reaching equilibria that correspond to the environmental situation. These equilibria of chemical signals finally affect the flagellum motor. If the bacterium is facing uphill in the gradient, the flagellum is moved in

CCW direction. In contrast, if the direction is not favourable, it is rotated in CW mode, thus is reorienting the bacterium. This way, the bacterium (averaged over many timesteps) moves faster in uphill direction than in all other directions, which gives it the needed bias to find its target areas in the environment. This integration of a spectrum of environmental stimuli and their memorisation over some time, as it is the case in chemicals in a cell, is assumed to be an important and easy way to achieve adaptive and directed behaviours. Resembling such systems in autonomous robots will perfectly fit to the small memories and to the low and noisy sensor capabilities of common swarm robots.

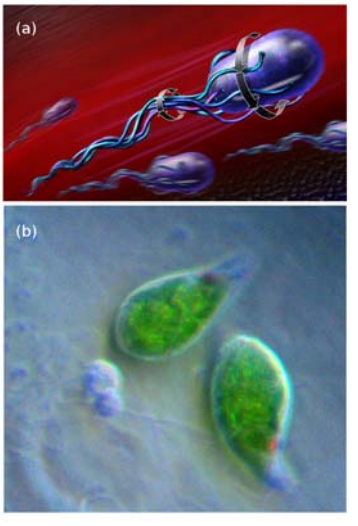


Fig. 4.7 (a) Bacteria of *Echerichia coli* navigate in environmental nutrient gradients by simple integration of chemical pathways linked to actuator modulation. For details see text. (b) The unicellular algae *Euglena gracilis* is able to navigate towards a light source by using just one single sensor device. For details, see text. Source of both pictures: Wikimedia Commons.

Also the unicellular algae *Euglena gracilis*, see Fig. 4.7, performs a comparable strategy (Bound & Tollin, 1967). It possesses a flagellum, a photoreceptor and a pigmenting shading unit (eye-spot), all combined together to one functional unit for photo-taxis. As *Euglena gracilis* moves, it rotates and the eye-spot shades the photoreceptor periodically. The periodic pattern of light and shade on the photoreceptor frequently switches the organism's behaviour between phobic reactions and straight swimming. In medium-intense luminance ($\leq 50lux$), this behaviour leads to positive photo-taxis, while in intense light ($\approx 500lux$, see (Häder *et al.*, 1981)) it leads to negative photo-taxis. This is important for efficient photosynthesis. Again in this example, it is chemical signal processing and spreading by diffusion that induces phobic reactions on the flagellum. Chemical signal transduction and processing allows a proper mapping of sensor input (eye spot and photoreceptor) onto the behavioural program of the actuator. In conclusion, this organism exploits body features (eye-spot) and feeds this embodied information into a chemical signal processing system. Again, this is probably a good inspiration for the inexpensive improvement of sensor precision and bias exploitation in swarm robotics.

Another algae *Volvox aureus* (Holmes, 1903) is living in spherical aggregates consisting of hundreds or thousands of singular algae cells, all equipped with flagellae, see Fig. 4.8. The pseudo-organisms (usually termed "colony" in biology) is able to perform collective photo-taxis to stay in favourable light conditions. This is achieved

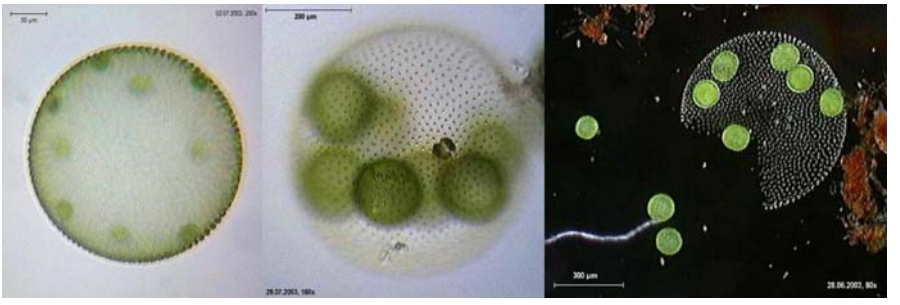


Fig. 4.8 Microscopical images of *Volvox aureus*. Each sphere is a colony of hundreds or thousands of cells. The spheres perform frequently photo-tactic movements to stay in favourable light conditions. Inside of the spheres, offspring (daughter spheres) can be seen. For details, see text. Source of all three pictures: Wikimedia Commons.

by modulating the flagellar activity of singular cells in a favourable way. Cells are ceasing their flagellar motion on the light exposed side of the spherical body while cells on the other side intensify their flagellar activity (Hand & Haupt, 1971). This pattern of flagellar activity turns the colony towards the light-stimulated side. It is also reported that the response to the light stimulus is stronger in the anterior region of the stimulated side (seen from the light source) compared to the rear side of the sphere, forming a gradient of flagellar activity. This gradient formation suggests that inter-cellular coordination, be it chemical, electrical or mechanical, might be involved in a further integration of environmental stimuli to allow colony-wide cell coordination. These colonies are not an organism, just like it was the case in the slime mould. Of course, they posses no colony-wide neuronal system, so cell coordination will be based on simple nearest-neighbour interactions of any kind. Again, it is a good source of inspiration for swarm robotics and for coordination principles in multi-modular robotics.

We propose that multi-modular robotic systems, especially if evolutionary computation (EC) will be used for shaping controllers, should draw inspiration from these evolutionary old micro-organisms. They demonstrate that simple reactive rules combined with chemical processes produce enough complexity to create complex behaviours. This way these organisms achieve high-level goals, even if they are aggregated to pseudo-organisms. The observed complexity in these systems is a product of self-organisation, that is based on simple networks of positive and negative feedback, coupled with delays and non-linear reactions. Our proposed hormone-inspired control system mimics those chemical processes that are either found in the form of cell-messengers in unicellular life forms or in the form of hormones in multi-cellular, higher life forms. In the following, we give a mathematical formulation of our proposed hormone-inspired robot control system, which was inspired by the above-mentioned chemical pathways in unicellular and multi-cellular life forms.

4.2.2 Related Work

The studies of (Braitenberg, 1984) demonstrated that a robot can be programmed to approach a target just by exploiting environmental gradients (e.g., light, sound, temperature). In cybernetics, the idea of homeostatic control of animal-inspired machines was an important component (Wiener, 1948). This theory demands for communication (see (Shannon, 1948), (Pierce, 1980)) of sensor values to central "control components". This component exerts positive and negative feedback via actuators back onto the future sensor input. The cybernetics approach focused on predicting and approaching future positions of mobile targets. Our hormone-inspired approach to control robots is inspired by the ideas of cybernetics, but expands it to a modern version of the "animat"-approach (McFarland & Bösser, 1993), which is aimed to mimic organisms. By involving evolutionary computation in shaping these "animats", we achieve a novel system of multi-modular robot control: AHHS.

Hormone-inspired control was suggested several times in literature for multi-modular robotics. Several studies suggest a hormone-inspired control system to coordinate movement of singular modules within a multi-modular robot organism. In these approaches hormones are more like messages (and/or hop-counts) that are routed among several robot modules in the CONRO system (Castano *et al.*, 2000), (Shen *et al.*, 2002) and in the SUPERBOT system (Hou & Shen, 2006a), (Hou & Shen, 2006b). In comparison to that, our AHHS hormones are modelled like fluid chemicals, which flow through a compartmentalised robotic organism. For example, in an AHHS the "conservation of mass" is a feature that is important. Another variant of hormone-inspired control was suggested by (Neal & Timmis, 2003) and by (Avila-García & Cañamero, 2005). Our hormone controllers regulate the behaviour of the robot without any other controller. In contrast, (Neal & Timmis, 2003) and (Avila-García & Cañamero, 2005) use virtual hormones to affect an underlying artificial neural Network. A different kind of hormone-inspired control is described by (Ogata & Sugano, 2000). A hormone-model was used to express "moods" of the robot and triggering of hormone secretion was affecting different pre-programmed and hand-coded behavioural controllers.

In the study of (Støy *et al.*, 2002), a multi-modular robot was controlled by identical programs on all robots, but behaviours of specific modules could be different, according to the position of the robot in the organism, as each of the modules picks out a subset of behaviours in accordance to its "role" in the organism. Also in our AHHS approach, all modules are driven by the same AHHS controller, initially derived from the same "genome". According to the past experience of the module and due to its specific sensor pattern on its position in the robotic organism, it possibly falls into specific modes of operation (roles).

The idea to coordinate robots by virtual gradients was used for body formation by (Stoy, 2006). To control robots in swarm mode, multiple approaches were published. The study of (Shen *et al.*, 2004) simulated the spread of virtual hormones to the local environment of robots. In biology such externally acting substances are called "pheromones". A "pheromone-based" approach to control a robot swarm was also published by (Payton *et al.*, 2001). Messages were routed among a robotic swarm

and hop-counts of messages were also considered by the robots to coordinate themselves. In (Schmickl & Crailsheim, 2007) the authors mimicked the behaviour of the slime mould *dictyostelium discoideum* to control a robot swarm by simulated chemical waves. The studies of (Schmickl & Crailsheim, 2008), (Schmickl *et al.*, 2007b), (Schmickl *et al.*, 2007c), and (Schmickl *et al.*, 2007a) used the exchange of virtual nectar (instead of virtual hormones) to control a robot swarm in various ways. A similarity to our system proposed here is that also the virtual nectar is subject to addition (secretion) and decay, thus is forming a homeostatic system. In contrast to our proposed AHHS, no evolutionary adaptation of rules, and thus no newly generated behaviours, emerge in these systems.

4.2.3 Artificial Homeostatic Hormone System (AHHS)

In the following, we describe the concept of AHHS (Artificial Homeostatic Hormone System) which mimics those evolutionary old controlling systems mentioned above. In our AHHS, a physiological model of the robot's inner body is controlling the robot's behaviours. Sensors trigger hormone excretions, which increase hormone concentrations in the robot's virtual inner body. These hormones diffuse, integrate, decay, interact and finally, affect actuators. To facilitate the emergence of complex behaviours, the virtual inner body is partitioned into several compartments, whereas each compartment is associated with a specific region of the real robot's body. This method of embodiment allows behaviours to act in the appropriate spatio-temporal context.

In our system, the change of hormone concentration H_i of hormone h in compartment c at time t is described by

$$\frac{\Delta H_h^c}{\Delta t} = \alpha_h + D_h \nabla^2 H_h^c(t) - \mu_h H_h^c(t) + \sum_i \mathscr{S}_i(t) + \sum_j \mathscr{H}_j(t), \tag{4.1}$$

for hormone specific production rate α_h, diffusion coefficient D_h, decay rate μ_h, and the summed influence of all applicable sensor rules $\mathscr{S}_i$ and all applicable hormone rules $\mathscr{H}_j$. The applicability of rules depends on the availability of sensor and actuators in compartment c. The numbers of rules and of hormones are defined through the genome. Hormones have maximal concentrations $H_h^{\max}$ that cannot be exceeded (i.e., $\forall h,t : 0 \leq H_h(t) \leq H_h^{\max}$).

The current actuator control value A of actuator a in time step t is defined by

$$A_a(t) = \begin{cases} A_a^{\min}, & \text{if } \sigma_a \sum_i \mathscr{A}_i(t) \leq A_a^{\min} \\ \sigma_a \sum_i \mathscr{A}_i(t), & \text{if } A_a^{\min} < \sigma_a \sum_i \mathscr{A}_i(t) < A_a^{\max} \\ A_a^{\max}, & \text{if } \sigma_a \sum_i \mathscr{A}_i(t) \geq A_a^{\max} \end{cases}, \tag{4.2}$$

for actuator rule $\mathscr{A}_i$, maximum actuator value $A_a^{\max}$, minimum actuator value $A_a^{\min}$, and actuator scaling constant σ_a that linearly scales hormone concentrations to

the relevant actuator control value interval. In the following, we define the three types of rules: sensor rule $\mathscr{S}$, actuator rule $\mathscr{A}$, and hormone rule $\mathscr{H}$. We define the sensor rule

$$\mathscr{S}_i(t) = \begin{cases} \sigma_s S_s(t)\lambda_i + \kappa_i & \text{if } \sigma_s S_s(t) \models_i \theta_i \\ 0 & \text{else} \end{cases}, \tag{4.3}$$

which is applied to a defined output hormone concentration H_h^c of hormone h in compartment c (cf. Eq. (4.1)), for sensor input $S_s(t)$ from sensor s, sensor scaling constant σ_s, dependent dose λ_i, fixed dose κ_i, and the trigger check for sensor threshold θ_i, and comparison operator $\models_i \in \{<, >, \lesseqgtr\}$ (whereas $\lesseqgtr$ represents the rule that triggers always).

We define the actuator rule

$$\mathscr{A}_i(t) = \begin{cases} H_k\lambda_i + \kappa_i & \text{if } H_k(t) \models_i \theta_i \\ 0 & \text{else} \end{cases}, \tag{4.4}$$

which is applied to a defined output actuator control value A_a of actuator a (cf. Eq. (4.2)), input hormone concentration H_k of hormone k, dependent dose λ_i, fixed dose κ_i, hormone concentration threshold θ_i, and comparison operator $\models_i$ defined as above.

We define the hormone rule

$$\mathscr{H}_i(t) = \begin{cases} H_k\lambda_i + \kappa_i & \text{if } H_k(t) \models_i \theta_i \\ 0 & \text{else} \end{cases}, \tag{4.5}$$

which is applied to a defined output hormone concentration H_h^c of hormone h in each compartment c (cf. Eq. (4.1)) and all parameters as defined above. Note that $h = k$ is allowed, thus self-referencing of a hormone is possible. This is essential to allow the emergence of feedback loops which in turn are crucial for self-organisation and homeostasis.

Hormones diffuse from all compartments to neighbouring compartments in our AHHS. They reach different concentration levels at equilibrium, see Fig. 4.9(a), because they are subject to decay in all compartments but are usually excreted just in some specific compartments. If, for example, just one sensor triggers a hormone secretion, the equilibria of this hormone will decrease spatially throughout the compartmentalised space. The further away compartments are from the triggering sensor, the lower is their steady-state hormone value. This way hormone values sometimes represent distance measures in the virtual body of the organism.

By linking compartments of several adjacent robotic modules, the diffusion process of AHHS is not restricted to a singular robot. In contrast, in the organism phase, the hormones flow throughout the whole organism, see Fig. 4.9(b). Hence, the same controller acts within the single free robot and within the joined multi-modular organism, similar to the case of *dictyostelium discoideum* aggregation, where chemo-taxis and cAMP excretion lead to different collective behaviours depending on the density of amoebas (singular moving, trail formation, slug

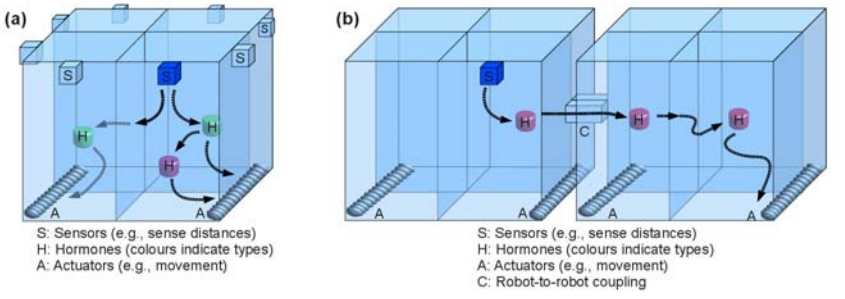


Fig. 4.9 (a) A sensor triggers excretion of a hormone in the right front compartment (sensor-rule). This higher hormone equilibrium modulates the actuator output on this side of the robot (actuator rule). It diffuses also to the left front compartment where it reaches a lower equilibrium, thus is modulating the left actuator to a lower extent. The sensor-excreted hormone interacts with another hormone (hormone-rule), which in turn affects the local actuator (actuator rule). (b) In organism phase, robotic modules are coupled. Through these couplings, hormones can diffuse and thus affect also distant actuators located on other robots.

movement, see (Camazine *et al.*, 2001)). The topology of the internal compartments, the physical properties of the hormones, as well as the network of hormone-to-hormone interactions is manifested (and evolved) in the artificial genome, which is an essential part of the AHHS. In the following, we give a formal description of the evolvable data structure that we call genome and that is used to parametrise the AHHS at starting time.

4.2.4 Encoding an AHHS into a Genome

Since evolution operates on a genome, a specific data structure, which parametrises the AHHS controller, is introduced. This data structure is called "genome" of the controller. It keeps the specific configuration of the controller persistent and it is the point of application of the artificial evolution.

The genome is a 2-tuple $\Gamma = (C_h, C_r)$ consisting of two logical entities: *hormone chromosome* C_h and *rule chromosome* C_r. If the self-organised compartmentalisation (see Sect. 4.2.5.1) is part of the evolution as well, we would have an additional chromosome C_m that encodes the morphogenesis-rules. There is one hormone gene G^H for each of the N hormones in the hormone chromosome $C_h = (G^H_1, G^H_2, \ldots, G^H_N)$ which is an N-tuple. There is one rule gene G^R for each of the M rules in the rule chromosome $C_r = (G^R_1, G^R_2, \ldots, G^R_M)$ which is an M-tuple. In Table 4.1 a listing of the genes in the two types of chromosomes is given. The hormone genes contain the actual parameters in a 5-tuple $G^H = (n, \alpha, \mu, D, H_{\max})$. The rule genes contain the actual parameters in a 9-tuple $G^R = (\models, T, \theta, \lambda, \kappa, s, a, h, k)$.

Table 4.1 The genome of the AHHS controller.

Gene	Description	Range
	Hormone Chromosome	
hormone ID n		$\mathbb{N}_0$
base production rate α	amount that is produced without sensory stimulation	$0 \leq \alpha \leq H_{\max}$
decay rate μ	cf. Eq. (4.1)	$0 \leq \mu \leq 1$
diffusion coefficient D	cf. Eq. (4.1)	$0 \leq D \leq 1$
maximum value of hormone $H_{\max}$	value at which a saturation is reached	$H_{\max} \in \mathbb{R}^+$
	Rule Chromosome	
Gene	Description	Range
rule comparison operator $\models$	partially defines the condition that is to be met for triggering action, see Eqs. (4.3) to (4.5)	$\models \in \{<, >, \lesseqgtr\}$
rule type T	type of triggered action ($\mathscr{I}$ represents the idle action)	$\{\mathscr{I}, \mathscr{S}, \mathscr{A}, \mathscr{H}\}$
threshold θ	partially defines the condition that is to be met for triggering action, see Eq. (4.3) to (4.5)	$0 \leq \theta \leq H_{\max}$
dependent dose λ	cf. Eq. (4.3), Eq. (4.4), and Eq. (4.5)	$0 \leq \lambda \leq H_{\max}$
fixed dose κ	cf. Eq. (4.3), Eq. (4.4), and Eq. (4.5)	$0 \leq \kappa \leq H_{\max}$
sensor input s	ID of the sensor that influences the hormone through a sensor rule $\mathscr{S}$	$\mathbb{N}_0$
actuator output a	ID of the actuator that is influenced by the hormone through an actuator rule $\mathscr{A}$	$\mathbb{N}_0$
hormone input h	ID of the hormone that is influenced through a sensor rule $\mathscr{S}$ or that influences another hormone through a hormone rule $\mathscr{H}$	$\mathbb{N}_0$
hormone output k	ID of the influenced hormone (through a hormone rule $\mathscr{H}$)	$\mathbb{N}_0$

4.2.5 Self-organised Compartmentalisation

The generation of appropriate compartment configurations is an important aspect of the AHHS. The compartment structure partially determines the space of potential behaviours of the controller. We present two variants of generating compartment topologies.

4.2.5.1 Compartmentalisation by Recursive Usage of AHHS

In the first approach to the compartmentalisation, we did not want to introduce additional functionality to the system to allow a dynamic and evolvable process that produces the compartment structure. Therefore, we developed a method to reuse many simple-configured AHHS in a specific manner to produce the compartment structure for a complex-configured AHHS. The advantage of this approach is that the same genome structure and the evolutionary operators can be used to evolve compartment structure and runtime behaviour of an AHHS.

We generate an additional set of morphogenesis-hormones and an additional set of morphogenesis-rules. These hormones and rules are similar to the rules described in the Sect. 4.2.3 and 4.2.4. The only difference is that they are used only at the initial start of the AHHS to generate compartments without having runtime-rules and runtime-hormones being computed. After the compartment-creation phase is over, no morphogenesis-rules and no morphogenesis-hormones are computed, as the AHHS switches to compute only runtime-rules and runtime-hormones.

To generate the compartments we start with a cube consisting of n^3 sub-cubes, each one being one AHHS (in the following, the term AHHS corresponds to these AHHS associated with a certain sub-cube) having just one compartment and a set of special morphogenesis-sensors which report the local morphogenesis-hormone concentrations. All robot sensors and actuators are associated to specific positions in this cube, which represent also their position on the real robot. This way, the grown compartments should allow embodiment of the robot morphology in the AHHS that is shaped by the process of compartmentalisation.

In analogy to Eq. (4.1), the concentration change ΔH_h of morphogenesis-hormone h in a AHHS at position (x,y,z) is defined by

$$\frac{\Delta H_h(t,x,y,z)}{\Delta t} = \alpha_h + D_h \nabla^2 H_h(t,x,y,z) - \mu_h H_h(t,x,y,z) + \sum_i \mathscr{M}_i(t), \qquad (4.6)$$

for production rate α_h, diffusion coefficient D_h, decay rate μ_h, and the summed influence of all morphogenesis-rules $\mathscr{M}_i$.

Each of these AHHS checks the set of available morphogenesis-rules, which either trigger a morphogenesis-hormone excretion ($H_j(t,x,y,z) \models_i \theta$ in analogy to Eq. (4.5)) or which trigger the growing of a compartment-wall at the location of the triggered AHHS (in analogy to Eq. (4.4), but with a binary effect of the triggering event, e.g., build wall or build no wall). Whenever a wall is built, one compartment is divided into two sub-compartments. The wall building action is implemented as a virtual actuator, the rest of the system acts similar to the description in Sect. 4.2.3. Another virtual actuator is merging of two compartments. If such an action is triggered by a morphogenesis-rule, the compartment, in which the triggering AHHS is currently located, is merged with one of its neighbouring compartments. The neighbour is chosen according to special rule parameters. It could be the oldest neighbouring compartment, the newest one, the biggest one or the smallest neighbouring compartment.

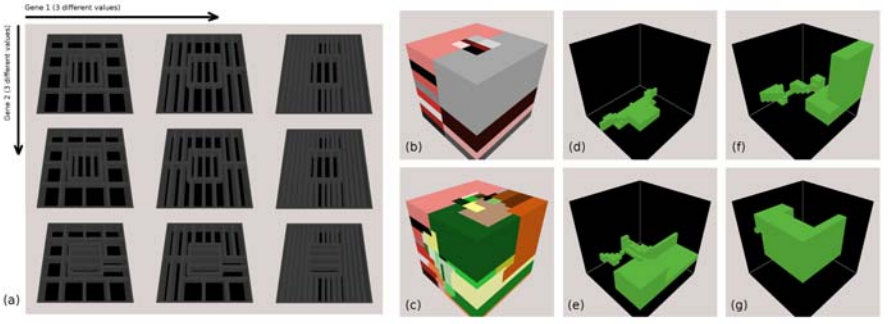


Fig. 4.10 (a) Layout variations resulting from combinations of two morphogenesis-rule mutations. The robot's internal layout was depicted by a two-dimensional system. Reprinted from (Schmickl *et al.*, 2009). (b) Compartments arising after 5 iterations over the set of the morphogenesis-rules. (c) After 22 iterations. (d)-(g): Shape of randomly picked compartments that emerge after 22 iterations.

After m steps of time, the process is stopped and the created compartment structure is parsed into a simple graph-shaped data structure, showing all neighbours for every compartment, compartment sizes and wall area of boundaries between neighbouring compartments. These information will later be used in calculating the diffusion of hormones among compartments and for calculating hormone concentrations. Sensors and actuators are linked to compartments according to their previously set position in the virtual space of the robot's body. At the end, all n^3 AHHSs are removed from memory, the explicit spatial representation of the robots inner space is removed from the robot's memory, as it was already compressed into the graph-like data structure mentioned above. After finishing the compartment-creating process, the compartmentalised AHHS starts to operate with processing runtime-hormone values and runtime-rules, as described in Sect. 4.2.3.

This process allows dynamic creation of compartment structure, which is well evolvable. It is able to generate many patterns by still keeping mutations similar to parental patterns in most cases. Fig. 4.10(a) shows how mutations of two variables in the genome affect the resulting compartment structure in a two-dimensional system. Fig. 4.10(b)-(g) demonstrates the high variability of compartment shapes in a three-dimensional system.

4.2.5.2 Compartmentalisation by Mimicking Slime Mould Behaviour

This second variant of generating compartment topologies was originally inspired by the behaviour of a slime mould species. This approach is based on an artificial self-organising multi-particle system consisting of reactive, mobile agents. These agents are virtual. The dynamics of this multi-particle system and the resulting compartment topology are computed offline before the AHHS-controlled robot is deployed.

The movements of the virtual agents, that constitute this multi-particle system, are governed by a few simple rules and interact indirectly via a virtual pheromone field. High values of this pheromone field are attractive to the agents. However, the agents are not allowed to turn with arbitrarily large angles. Instead, they can only turn with a defined angle velocity. Based on a few parameters, that are easily integrated into the genome, the system generates a wide variety of complex patterns. Some of these patterns show a notable property: seemingly never-ending, high dynamics in forming and reconfiguring complex patterns. For appropriate parameters, the multi-particle system generates network-like patterns that are converted into compartment configurations. This method of generating compartment configurations is assumed to be easily evolvable.

In the following, we define the multi-particle system as introduced in (Hamann, 2009; Jones, 2009). The agents move in two-dimensional space (generalisation to 3-d is straight forward) with periodic boundary conditions (torus). The change of an agent's position $\mathbf{x}$ is defined by

$$\frac{d\mathbf{x}}{dt} = \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} v, \tag{4.7}$$

for a constant velocity $v > 0$, except for the case of a local pheromone value that is bigger than a threshold P_{max}, then we set $v = 0$. The change of the agent's direction ϕ is defined by

$$\frac{d\phi}{dt} = \alpha(s_l(t), s_c(t), s_r(t))\gamma(t), \tag{4.8}$$

for $\alpha(s_l(t), s_c(t), s_r(t)) \in \{1, 0, -1\}$ defining the direction of the turns (clockwise, no turn, or counterclockwise), γ defining the absolute value of the turn angles, and for sensor values s_c and s_l that are defined by

$$s_c(t) = P\begin{pmatrix} x_1 + \cos(\phi)d \\ x_2 + \sin(\phi)d \end{pmatrix}, \tag{4.9}$$

$$s_l(t) = P\begin{pmatrix} x_1 + \cos(\phi - \psi)d \\ x_2 + \sin(\phi - \psi)d \end{pmatrix}, \tag{4.10}$$

for a pheromone field P, $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, sensor angle ψ, and sensor distance d. s_r is defined analog to s_l. Closely following (Jones, 2009), we define the process that determines whether there is a turn and its direction

$$\alpha(s_l(t), s_c(t), s_r(t)) = \begin{cases} 0, & \text{for } s_c(t) > s_l(t) \\ & \wedge s_c(t) > s_r(t) \text{ (no turn)} \\ \pm 1, & \text{for } s_c(t) < s_l(t) \\ & \wedge s_c(t) < s_r(t) \text{ (random turn)} \\ +1, & \text{for } s_l(t) < s_r(t) \text{ (right turn)} \\ -1, & \text{for } s_r(t) < s_l(t) \text{ (left turn)} \end{cases}, \tag{4.11}$$

whereas the order (top to bottom) of the conditions matters. The random turn has a probability of 50% for $+1$ and 50% for -1. We define the absolute turning angle

$$\gamma(t) = \begin{cases} \phi_{\text{rot}}, & \text{for } t \in \{0, \tau, 2\tau, \dots\} \\ 0, & \text{else} \end{cases}, \tag{4.12}$$

for a constant rotation angle ϕ_{rot} and a time interval τ at which the agents turn and their directions are updated (typically $\tau = 1$). Thus, we obtain a synchronised system that is discrete in time. The pheromone field P is, in principle, defined by the standard diffusion equation

$$\frac{\partial P(\mathbf{x},t)}{\partial t} = D\nabla^2 P(\mathbf{x},t) - \eta P(\mathbf{x},t) + \theta \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i(t)), \tag{4.13}$$

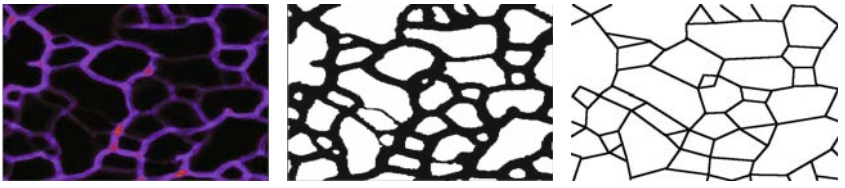
for diffusion D, evaporation rate η, addition θ (the Dirac delta δ indicates that an agent only contributes to the pheromone field at its position), number of agents N, and agent positions $\mathbf{x}_i(t)$. However, for simplicity and to reduce the computational complexity the diffusion and the evaporation are only executed at $t \in \{0, 10\tau, 20\tau, \dots\}$ unlike the addition process that is executed at $t \in \{0, \tau, 2\tau, \dots\}$.

The patterns, that are generated by this system, show a high variety depending on the used parameters (Hamann, 2009). Only those that form erratic and network-like structures are relevant here. An appropriate parameter setting is given in Table 4.2.

A typical pheromone field generated by this system is shown in Fig. 4.11(a) (higher pheromone intensities are shown in warmer colours). Using a simple threshold method, the pheromone field is reduced to a binary image, see Fig. 4.11(b). The underlying compartment topology is identified by applying edge detection, see Fig. 4.11(c).

Table 4.2 Used parameters.

Parameter	Value
sensor angle ψ	45^o
rotation angle ϕ_{rot}	45^o
rotation period τ	1 [time units]
sensor distance d	0.01 [length units]
velocity v	0.01 [length units$/\tau$]
diffusion D	0.05 $[(1/L)/(10\tau)]$
evaporation η	0.06 $[1/(10\tau)]$
addition θ	5 $[1/\tau]$
number of agents N	700
max. pheromone value P_{max}	300
simulated steps	1×10^3 [time units]



(a) Pheromone field. (b) Binary image. (c) Underlying compartment topology.

Fig. 4.11 Extraction of the compartment topology out of the pheromone field. Using a threshold the pheromone field (a) is reduced to a binary image (b). Based on this image the underlying topology (c) is extracted through edge detection.

4.2.6 Evolutionary Adaptation

In order to synthesise AHHS controllers for any desired behaviour we apply evolutionary algorithms. That is, we start with randomly generated AHHS controllers that are evaluated, selected, mutated, and recombinated. We restrict ourselves to offline evolution (i.e., controllers are evolved before the robots are deployed), since this process typically needs many generations to evolve useful behaviour. The evaluation of the behaviour generated by the AHHS controllers is done using the Symbricator3D Simulation (see Sect. 2.4.2). This standard evolutionary approach is shown in Fig. 4.12.

The application of a recombination is determined by a stochastic process and the defined recombination probability. The recombination operator is a standard two-point crossover, whereas the description of the hormones and the description of the rules are treated as two independent chromosomes (resulting in two two-point crossovers).

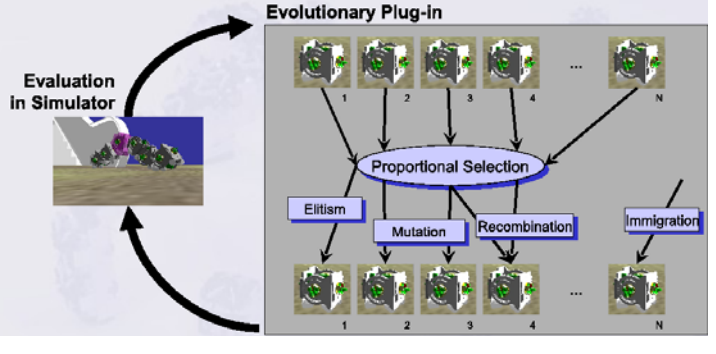


Fig. 4.12 The evolutionary loop: evaluation of population, selection, and generating the new population through elitism, mutation, recombination, and immigration.

In contrast, the definition of the mutation operator is more relevant and more interesting. In fact, there are two independent mutation operators – one for each chromosome type.

The mutation operator acting on the genes of the hormone chromosome changes with a probability P_H one of the parameters $p \in \{\mu, H_{\max}, \alpha\}$ (decay rate μ, maximum hormone value $H_{\max}$, and base production rate α). The hormone ID and also the diffusion coefficient are usually not varied. The mutated parameter p' is ensured to stay within a defined interval $p' \in [(1-m)p, (1+m)p]$, for a mutation range parameter $m \in [0,1]$. Hence, the hormone gene mutation operator in pseudo-code notation is defined by:

Algorithm 1. Mutation operator for the hormone gene.

```
foreach hormone gene G^H in hormone chromosome C_h do
    r ← RandomUniform(1)
    if r < P_H then
        p ← ChooseOne({μ, H_max, α})
        if RandomUniform(1) < 0.5 then
            sign ← +1
        else
            sign ← -1
        end
        change ← RandomUniform(m)
        p ← (1 + sign * change) * p
    end
end
```

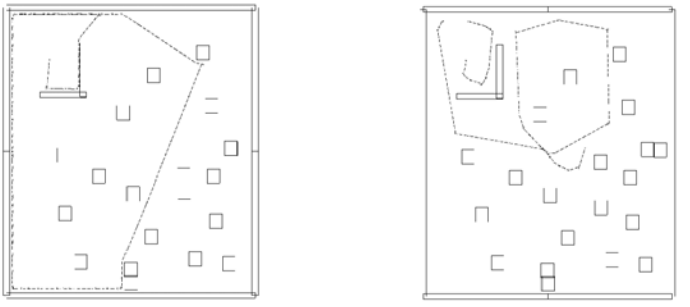
whereas `randomUniform(a)` returns a random number out of $[0,a]$ based on a uniform distribution. In addition, the resulting values are checked to stay within certain bounds as indicated in Table 4.1 (usually $H_{\max}$ would also be bounded).

The mutation operator acting on the genes of the rule chromosome is analog to the mutation of hormones and changes with probability P_R one of the parameters $p \in \{\models, T, \theta, \lambda, \kappa\}$. Again the mutation range parameter m is applied to the continuously defined variables. In case of the discrete valued rule comparison operator $\models$ and rule type T, one of the possible values is picked randomly (e.g., $T = \mathscr{S}$ might change to $T = \mathscr{I}$, $T = \mathscr{A}$, or $T = \mathscr{H}$).

Furthermore, the number of rules could be changed by addition or deletion of rule chromosomes. Changing the number of hormones is more complex because the rule chromosomes include hormone IDs. Hence, the case of rules, that refer to a deleted hormone, would need a clarification beforehand.

4.2.7 Single Robots

The feasibility of the evolutionary algorithm applied on the AHHS controller was tested on single robots. In a simple "explore the environment" task the initially



(a) (b)

Fig. 4.13 Trajectories of the robot of two different evolutionary runs in an environment with walls. Reprint from (Stradner *et al.*, 2009).

randomly configured controllers were optimised in order to activate the screws (see Sect. 2.1.3) of the robot correctly to cover some distance. A further increase of the fitness value was achieved, if the controller learned to react on sensor inputs. By recognising walls and avoiding or following them a better performance could be achieved. Thus, the fitness function was chosen in that way that moving and gaining distance from the starting point led to an increased fitness value.

The evolvability of the AHHS controller could be shown by the fact that the task described above was solved after at most 50 generations. As a result of these experiments different motion patterns were observed. For example two different patterns are shown in Fig. 4.13: wall follower and wall avoider.

A closer look on the evolved hormones and rules reveals the mechanism of the controller that steers the robot in a post evaluation. Three hormones are responsible for the motion behaviour of the robot (Fig. 4.14(a)): Two hormones are insensitive on sensor input and reach different equilibria ($H0 \rightarrow 9$, $H2 \rightarrow 84$, see Fig. 4.14(b)), during run time. In connection with the third hormone, $H7$, a different driving behaviour occurs depending on whether a wall is nearby. In the absence of obstacles, a straight trajectory is accomplished because of the simultaneous activation of actuator $A0$ by $H7$ and $H2$. Only this combined activation is strong enough to match the activation of actuator $A1$ by $H0$ for straight driving. In the case of sensor $S3$ detects an obstacle, the production of hormone $H7$ ceases and the different activation of the two screws leads to a left turn. Thus, even in this simple task an already complex network consisting of three hormones and 4 rules was evolved by the evolutionary algorithm.

The dynamics of the hormone concentrations which generate the behaviour of the wall follower and the wall avoider are plotted in Fig. 4.15.

4.2.8 Forming Robot Organisms

The AHHS controller is not only responsible for the behaviour of the single autonomous robot modules but also for the robot organism. One major point of this

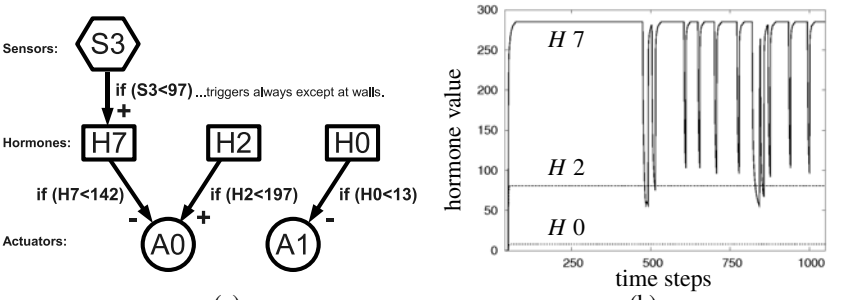
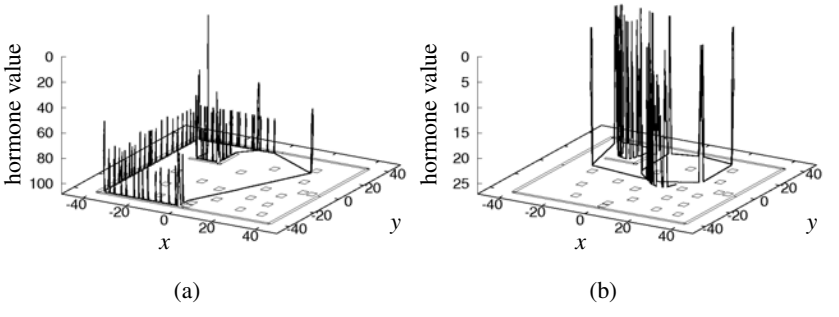


Fig. 4.14 (a) Schematic overview of the sensor-hormone-actuator interaction in the wall follower controller (see Fig. 4.13(a)). (b) Values of the three participating hormones. Reprint from (Stradner *et al.*, 2009).



(a) (b)

Fig. 4.15 The value of the critical hormone in (a) the wall follower controller (compare Fig. 4.13(a)) and (b) the wall avoider controller (compare Fig. 4.13(b)). Reprint from (Stradner *et al.*, 2009).

topic is the formation and reconfiguration of the robot organisms. To keep the morphology of the robot flexible, the AHHS should be able to perform this task in a self-organised manner. Because robustness of a robot organism is often in contrast to flexibility, the trade-off between these two features must be balanced and resolved.

Starting to build a robot organism out of a swarm of single modules and the reconfiguration of an existing robot organism are similar processes. In both processes an additional number of nearby single robot modules must be available. Furthermore, a trigger from the environment leads to an addition or removing of modules or a change of the morphology of the organism body shape. This trigger event has to be perceived by one of the modules. Then it has to be communicated externally to other single modules nearby or internally to the other robots of the organism in a self-organised manner through hormone value changes.

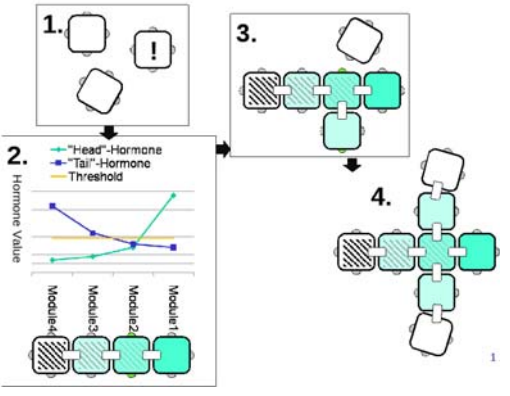


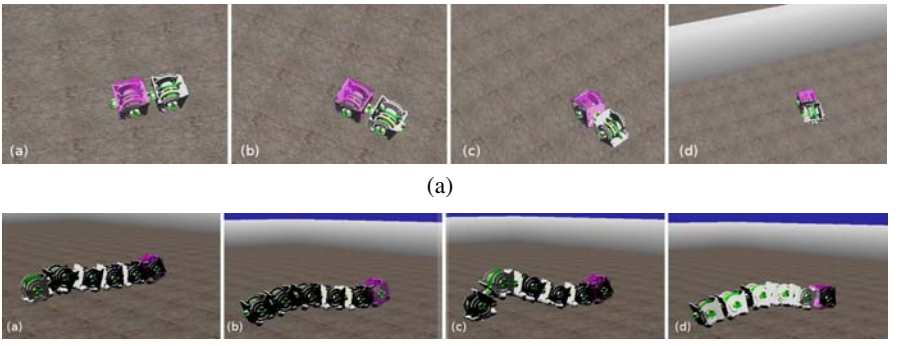
Fig. 4.16 The development of the body formation. The process of the progress from single module formation in a swarm to robot organism with legs is depicted in four steps. One possible way of achieving this with AHHS is denoted as a schematic graph of hormone values of two hormones in step 2. For further explanation see text. Reprint from (Schmickl *et al.*, 2009).

An example of a body formation process is depicted in Fig. 4.16. The module, which detects the environmental trigger, changes its state which in turn is a trigger for the nearby modules for docking. After the docking process is finished the hormones diffuse between the single modules inside the organism. Starting from one module different shapes depending on the gradients of the hormones can be built. Rules, which underlie artificial evolution, define thresholds, for example, where limbs should start to grow (see Fig. 4.16, step 2 and 3). Self-organisation of the morphology of the robot organism is established by changing these rules through evolution, learning or by a new occurring environmental trigger.

4.2.9 Locomotion of Robot Organisms

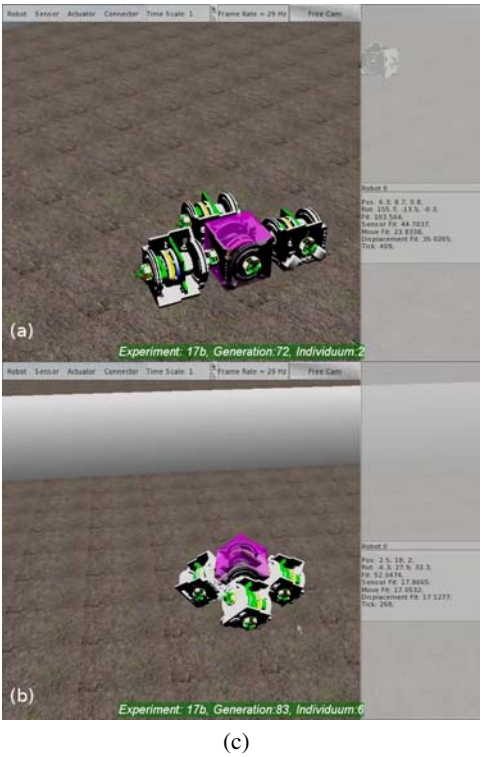
After we tested the AHHS on singular moving robots using various types of locomotion (differential drive, screw drive), we wanted to test the system also in organisms of coupled autonomous modules. We introduced a randomised AHHS with 15 randomly parametrised hormones and with 60 randomly parametrised rules, 12 distance sensors and 1 hinge as the only actuator. The hinge is generally not able to move a single robot module, as it just bends the module. A single module will just fall to the side after a significant hinge contraction. In contrast, two joined modules that oscillate their hinges in a synchronised way, are potentially able to move in the environment efficiently. We allowed the virtual hormones to diffuse through the robot-to-robot connectors from each robot module to its neighbouring modules, as it is depicted in Fig. 4.9. This way sensory input and hormone secretion in one module could affect also hormones and actuators in other modules, establishing an organism-wide hormone-based communication system.

We initially coupled two robot modules in the Symbricator3D Simulation, see Sect. 2.4.2, and allowed it to perform 1000 time steps. Those organisms that moved the longest distance in the arena were selected for seeding the next generation (population size: 20 organisms). Before that they were recombined and mutated. In several experiments we encountered the first moving organisms within a few generations.



(a)

(b)



(c)

Fig. 4.17 Stroboscopic snapshots of an organism consisting of: **(a)** two autonomous modules. The only locomotion that was possible was periodic bending of the modules by modulating the hinge-angle; **(b)** six autonomous modules. With this body form, the robots exhibited a caterpillar-like motion pattern, which was evolved from a randomly initialised AHHS controller; **(c)** four autonomous modules in a T-shaped organism. With this body form, two distinct forms of motion evolved. (top) Synchronised oscillation of modules in a flat body constitution. (down) Leg-like walking in an erected body constitution. Reprints from (Schmickl *et al.*, 2009).

After approximately 20 generations we always found several fast moving organisms, see Fig. 4.17(a), in the population.

After this was observed, we added another module to the organism and continued our artificial evolution. Initially, the change of body form slowed down the organisms significantly, but after additional 10-15 generations they reached almost

the same motion speed again. After this AHHS was evolved, we added again three modules. Within 20-30 generations, these larger organisms evolved an effective and well coordinated motion pattern. Hinge contractions moved like a travelling wave from the tail to the front, resulting in a caterpillar-like movement, see Fig. 4.17(b). The head module in front was most of the time erected, minimising friction and supporting the organism's motion this way.

Finally, we investigated our AHHS oscillator that was initially evolved in the two-module setup in a more complex body shape. Four modules were connected in a T-shape manner. During the course of evolution two distinct motion strategies evolved. First, a behaviour of synchronised contraction of hinges evolved that was similar to the caterpillar-motion but adapted to the distinct body shape. After some time, a totally different behaviour emerged from evolution. The central unit in the T-shape body contracted permanently, this way erecting the whole body. The three branch-modules then contracted in an oscillatory manner, moving the whole organism like a walking tri-pod, see Fig. 4.17(c).

In all of the body shapes that we tested, artificial evolution was able to produce novel motion patterns that were well adapted to the body morphology and to the surrounding environment. We observed that it is significantly more efficient to evolve body shape together with body control. Thus, a dynamic and genetically adaptable method of body formation is assumed to be advantageous, as both adaptations can be performed in the same evolutionary loop using the same genome.

4.2.10 Feedbacks

Feedback loops (Fig. 4.18) emerge within the functionality of the single robot modules and the robot organism and the artificial evolution. Some of these feedback loops are found in "classical" evolutionary robotics concerning single robots, others exist in non-evolutionary multi-modular robotics. In the context of AHHS and

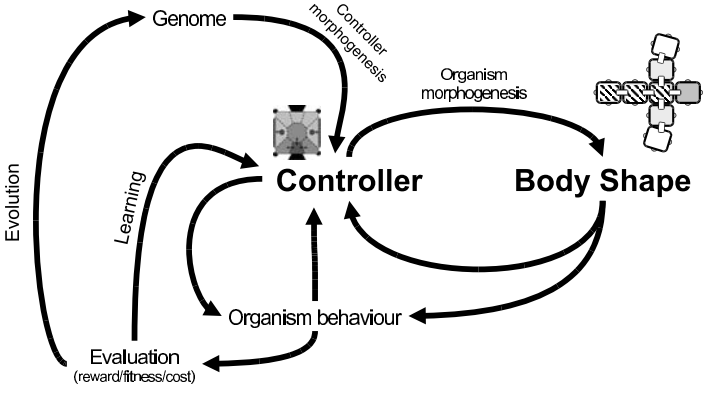


Fig. 4.18 The feedback loops that affect the evolution of the AHHS controller and organism shapes. Reprint from (Schmickl *et al.*, 2009).

their ability to control the single modules and joined organisms, these loops interact with each other. This interaction plays a crucial role in the pluripotent function of the AHHS.

Six feedback loops can be identified: classic control, learning, evolution, controller morphogenesis, robot organism morphogenesis, and body motion; for details, see (Schmickl *et al.*, 2009). All six feedback loops can be implemented in real robotic hardware and in a sophisticated simulation software and they can be tested separately. For example, by using hand-coded body shapes the motion behaviour of the robot organism can be investigated. In case of learning and evolution, it depends on the situation whether both of them are necessary in the system. They only differ more or less in their time scales. The duration of learning lasts for one life time while evolution is focused on generations. For example, there might be scenarios in which the performance during runtime cannot be improved significantly such that no optimisation by learning is needed. However, all feedback loops interact with each other in a complex way which is one of the key points of this approach. These intertwined feedback loops encourage and challenge evolution to generate adaptive behaviour.

4.2.11 Conclusion

AHHS controls the behaviours of robots in a way that allows achieving internal homeostasis of intrinsic hormone levels. In the simple case of collision-avoidance this is achieved by sensor-rules that disturb homeostatic set-points by triggering additional hormone secretion as the robot approaches an obstacle. Already in cybernetics filtering of sensory input was an important aspect. The hormone values in our AHHS integrate past sensory inputs over time. Another important feature of our AHHS is the steady decay of such integrated information. This feature allows "forgetting" of outdated information.

Another novel feature of our controller is the compartmentalisation of our robot's virtual inner space, which allows embodiment of the controller. Sensors are linked to specific compartments and trigger hormone secretion only there. Compartments allow spatial computation of sensory input. In the joined robot organism, whole robot modules support this compartmentalisation, reacting to their local hormone values and supporting embodiment on the organism level this way.

As mentioned above the AHHS controller shows inherent homeostatic processes. This tendency towards moderate values is in contrast to other controller types such as artificial neural networks which sometimes tend towards extreme values. Another advantage of the AHHS is that its functionality is the same for a single robot module as in the case of the whole robot organism. There is no difference between the diffusion based on compartments inside a robot module and the diffusion based on whole robot modules. Such a change of topology would most likely result in a drop of performance in case of standard robot controllers. For most other kinds of robot controllers, two classes of controllers would be needed to be produced: one controlling a single module and one controlling the whole robot organism. The verification

of these assumed advantages of the AHHS including the above mentioned will be a main objective of our future work.

4.3 Evolving Artificial Neural Networks and Artificial Embryology

Ronald Thenius, Michael Bodi, Thomas Schmickl, Karl Crailsheim

In contrast to the above mentioned controller, Artificial Neural Networks (ANNs), are used since decades, also for autonomous robots. A rather novel approach is to combine such standard ANNs with a virtual internal neuromodulator system, which allows regulation of a whole ANN via the linkage of sensors of the robot to virtual neuromodulator glands. Many parameters of the morphological structure of the neural network can be shaped by a simulated embryological process at startup time of the robot (after booting the robot). Such embryological processes can be tuned by various evolutionary computation methods. The resulting controllers are called "Evolving Embryogenetic Brainstructures" ($\varepsilon\varepsilon$B)

For $\varepsilon\varepsilon$B controllers, the life-cycle of the robotic units (of a robotic swarm) can be broken down into the following 4 phases:

Initial phase of the robot: The $\varepsilon\varepsilon$B, that controls the robot at runtime, develops during the embryological process. Depending on the calculation power and memory available onboard the robot, the neural network is then uploaded to the robot at the beginning of its lifetime, or calculated directly onboard the robot. The network is then executed by an ANN interpreter running on the robot.

Swarm phase: While Robots are moving around 'alone' in the arena, and are not linked to other robots, they perform basic tasks like data collection or exploring the area. During this phase, the neural network is already shaped by embryogenesis, but is still able to be adapted by learning techniques. Under certain conditions, for which the controlling neural network is trained or evolved for, the robots start to aggregate. The reason for such an aggregation can be: linkage of robots to use faster inter-robot communication channels, the exchange of energy via an energy bus, or the forming of a higher level organism.

Body formation: The decision about the bodyshape to form is an important one and will be shaped by training or evolution to the environments (or conditions) the swarm is intended to work in. The robots have to arrange themself into spatial patterns and call predefined routines for linking to its neighbours.

Organism phase: If robots are linked together in one big robotic organism, the $\varepsilon\varepsilon$Bs of the robots are able to communicate through direct neuron-to-neuron interfaces, which are established by the internal communication bus of the organism. Still each unit is responsible for its own motors and the processing and passing through of sensory information. A special case of the organism phase is when small groups of robots have linked to small autonomous organisms, thus when swarms of relatively simple organisms start to explore the area. This might be advantageous to increase the search radius of the whole robot swarm.

We simulate virtual embryologic processes to allow the $\varepsilon\varepsilon$B-controller to "grow" in a structured way. Many parameters of the $\varepsilon\varepsilon$B-controller are shaped by these processes: the number of cells, their degree of connectivity to their neighbours, and the spatial distance between clusters of highly connected nodes. We use this novel method to organise the growth of the embryo, that later form the neural network. As guide for this development we used processes that are observable in biology during the developmental phase of most multicellular lifeforms. The growth process of the virtual embryo is controlled by a genome that is defining the reaction of a cell under certain conditions during the developmental phase.

Building an evolutionary system enables us to shape controllers. It is important to find or develop an "evolution friendly" method. For the approach described in the following, we chose the biological process of embryogenesis as a model. During biological evolution the processes of embryogenesis showed to be an ideal tool for shaping the bodies (including the control structures) of all multicellular lifeforms (Carroll, 2006). The mechanisms working within an embryo (known as "EvoDevo") are perfectly able to work as a substrate for evolutionary processes. In biology, four phases can be observed in the evolution of multicellular organisms:

1. **Optimisation:** A morphometrical structure (including controlling structures, such as neural ganglia) are optimised by evolutionary processes.
2. **Serialisation:** Due to changes on genetic level the optimised morphological structure is built identically several times within an animal.
3. **Exploration:** Former identical morphological structures start to differ in their shape and function. Because of the presence of functioning copies of these structures, the evolutionary pressure on a single functional unit is low. In this way even disadvantageous changes in shape and function of a single functional unit are not punished by evolution. In this way "fitness-valleys" can be overcome.
4. **Specialisation:** If a new shape becomes advantageous for a new function, the optimisation process is repeated. Disadvantageous structures are discarded.

One process comparable to EvoDevo, found in technical optimisation methods, is "simulated annealing", whereby, in opposite to the biological analogon, the changes in the fitness function are induced externally, and do not arise from within the system.

The method of morphological shaping of a controller is interesting for a wide field of controller types: besides the ANNs, which are the main focus of this section, other controllers that also include a morphological component, like the Artificial Homeostatic Hormone controller (AHHS, (Schmickl & Crailsheim, 2009)) or GasNets can gain advantages from the ideas of morphological shaping by artificial embryogenesis.

4.3.1 Shaping of ANN in Literature

Structured ANNs can provide highly interesting auto-teaching structures as mentioned by Nolfi in (Nolfi & Parisi, 1993) and (Nolfi & Parisi, 1997). "Auto-teaching

networks" consist of two subnets, a "teaching network" and a "controlling network". The authors describe, that such a network, if it is shaped by an artificial evolutionary process, has "genetically inherited predispositions to learn". The virtual embryogenetic approach described in this section aims for the evolution of neural networks with substructures with this ability.

Other concepts of the influence of bodyshape (and neural controller shape) to the function of the controller itself and to the control process are described in (Pfeifer & Bongard, 2006) and (Pfeifer *et al.*, 2005b). The main idea of these publications is the intense influence of the morphological shape of the agent (or robot) on the learning and controlling process. As described later in this section, the positions of sensor input interfaces and actuator output interfaces within the virtual embryo's growing space represent the positions of sensors and actuators on the real robot. This way, we expect advantageous effects, e.g., from the accumulation of sensory input interfaces on one side of the embryo.

Other interesting approaches to the problem of shaping ANNs (with focus on the French flag problem described in (Wolpert, 1998)) are described in (Miller & Banzhaf, 2003). In this work, the function of a node is not determined, but can be shaped by the genome. Another work that deals with the problem of differing functions within an ANN is (Timmis *et al.*, 2009), in which different predispositions for learning are implemented by "virtual adaptive neuro-endocrinology". Within such a network, different types of cells exist: gland-cells, which influence the learning of the network, and regular cells, which are influenced by the gland-cells.

A different approach to shape the structure of an ANN has been described in (Hampton & Adami, 2004). In the system described, the position of the cell is fixed from the beginning, the growth of axons is controlled by morphogenes diffusing throughout the embryo. The advantages of this system seem to be the higher tendency for self-similarity of network structures during evolutionary processes, but on the other hand it seems that it has a low probability to develop differentiating cell-types with different functions.

A very detailed model of artificial embryogenesis is described by Bongard in (Bongard & Pfeifer, 2003). In this work, 16 genetic commands are defined that are needed to build an embryo and link neural cells within this embryo. These 16 commands are comparable to the genetic commands described in this section (for a simplified list see 4.3.5). The main difference between the work described in this chapter and the work of Bongard is the possibility of the development of non-neural cells, which are advantageous for morphological shaping of the embryo (for details see Sect. 4.3.8). State-of-the-art neuro-evolutionary techniques are also described in (Floreano *et al.*, 2008b). Other inspiration for our work comes from the field of evolving electrical circuits (e.g., (Mattiussi & Floreano, 2007)), in which electrical devices (like resistors, capacitors or ICs) and their position in a circuit are coded in a genome and evolved.

Models of developmental processes of embryos in nature have been investigated since decades. First ideas about possible self-organisation processes shaping or structuring a living creature can be found in (Turing, 1952), where

the authors describe the interaction of different antagonistic chemical substances diffusing in a medium. Early models investigating growth-processes, repair mechanisms and structuring processes of organisms (mostly on simple live forms like *Hydra sp.*) were published in (Gierer & Meinhardt, 1972) and (Babloyantz & Hiernaux, 1974), an interesting hydra model was published in (Meinhardt & Gierer, 2000). Early concepts of pattern formation in animals and embryos are described in (Gierer & Meinhardt, 1972). First discussions of mechanisms controlling the growth processes in insects are made by Kalthoff (Kalthoff, 1978). This field of research is called evolutionary developmental biology, also known as "EvoDevo". A good overview over the field of EvoDevo can be found in (Müller, 2007) and (Carroll, 2006).

One important field of research, influencing the work described in this section, is the modelling of repair mechanisms of cell-clusters. The work done on this topic is not primarily aiming at the shaping of ANNs, but at dealing with problems like coordination of groups of cells, especially the control of growth processes (Basanta *et al.*, 2008; Andersen *et al.*, 2009), which helps stabilising the growth of virtual embryos and helps determining the size of ANNs.

4.3.2 Overview over Section

In this section, we describe how the organisation of nodes and links of an ANN using virtual embryogenesis is implemented. In our concept of virtual embryogenesis, which is described here, we are mimicking processes observable in biology during the developmental phase of most multicellular life-forms, like *Drosophila m.* or other species (Jaeger *et al.*, 2004; Meinhardt & Gierer, 2000; Babloyantz & Hiernaux, 1974). We also included general concepts of biological embryogenesis and artificial embryogenesis (Wolpert, 1969; Basanta *et al.*, 2008) into our concept. Complex processes, like physics or diffusion, are strongly abstracted in our virtual embryogenesis to enable a later optimisation of the resulting $\varepsilon\varepsilon$B, by using artificial evolutionary processes, see Fig. 4.19. The fast calculation of single embryological processes is necessary, due to the requirement for fast simulation of embryological processes on systems with limited hardware resources, especially for projects dealing with evolution in autonomous robotic systems, such as SYMBRION/REPLICATOR.

4.3.3 Concept of Adapting Virtual Embryogenesis for Controller Development

In our concept of virtual embryogenesis the embryo consists of individual cells. Within the developmental process cells can become nodes in the ANN. The cells of our virtual embryo can duplicate, die, specialise, emit morphogenes (chemical substances that can diffuse throughout the whole embryo), or build neural links to other cells in the embryo. An important aspect that we implemented in our model, is that a cell has no ability for active movement, but can be "pushed around" in

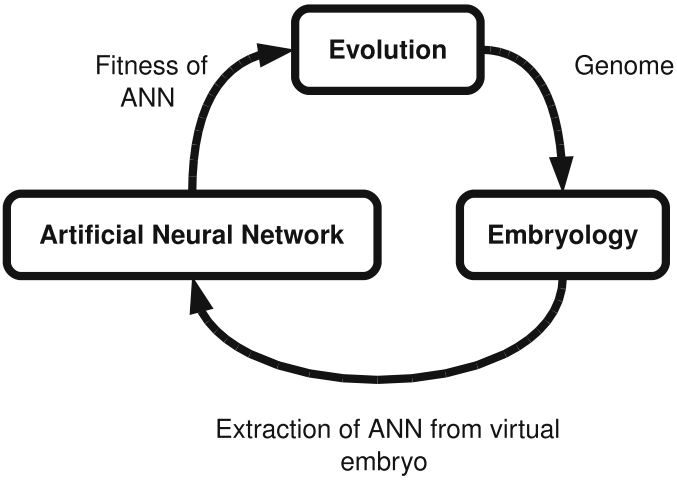


Fig. 4.19 Optimisation of an ANN using artificial evolutionary processes and artificial embryogenesis. For details please see 4.3.15. Reprinted from (Thenius *et al.*, 2009).

space, due to growth processes (duplication processes of other cells). The genome of a cell defines the cells' actions. Genes (subunits of the genome) can be triggered by virtual morphogenes. Due to the ability of genes to produce other morphogenes if activated, a network of feedbacks emerges. The growth of the embryo (and the embedded neural network) is governed by the resulting self-organised process. As soon as the embryological process is finished, the developed network is analysed, translated into a data structure which is compatible to a standard ANN-interpreter, and tested in a simulation.

4.3.4 Diffusion Processes

We implemented the embryo in our model as a multi-agent model. A single cell is represented by an agent. The space and time in our model is discrete. The time step is indicated by Δt. Each spatial unit (patch) can be occupied by only one cell. The cells interact with each other by virtual physics and virtual chemistry. Cells are able to emit morphogenes, which diffuse throughout the embryo (Crick, 1970). The concentration $c(m,x,y,t)$ of a morphogene m at the position (x,y) at time step t is calculated according to Eq. (4.14), whereby $c_{\max}(m)$ is the maximum concentration of the morphogene m. The maximum concentration $c^{\mathcal{N}}_{\max}(m,x,y,t)$ of the morphogene m in the cell z at the position (x,y) and in all neighbouring cells ($\mathcal{N}$ denoting the Von Neumann neighbourhood and the cell itself) at the time step t is calculated according to Eq. (4.15). The amount of the decrease of the morphogene concentration when diffusing from one patch to another is $d(m)$. When a cell at position x,y emits a morphogene, its value $c(m,x,y,t)$ is set according to Eq. (4.16). Please note that no conservation of mass is implemented in our model. This simplification of

real physical diffusion processes is necessary to achieve the required computational speed. The result of this abstract diffusion model suffices for our needs to achieve the desired embryogenesis.

$$c(m,x,y,t) = \min(c_{\max}(m), c^{\mathcal{N}}_{\max}(m,x,y,t-\Delta t) - (d(m) * \Delta t)), \tag{4.14}$$

$$c^{\mathcal{N}}_{\max}(m,x,y,t) = \max_{(x,y)\in\mathcal{N}}(c(m,x,y,t)). \tag{4.15}$$

$$c(m,x,y,t) = c_{\max}(m). \tag{4.16}$$

4.3.5 Genetics and Cellular Behaviour

A virtual cell z with the position (x,y) in our model measures the concentrations of morphogenes every time step and reacts in a pre-programmed way. The type of the reaction to the presence of a moprphogene as well as the measured concentration $k(m,z,t)$ of a morphogene m in the timestep t, needed to trigger the reaction is specified in the genome of the cell. The genome Γ is a set of N genes G (see Eq. (4.17)), whereby each gene is a tuple of numeric values (see Eq. (4.18)). The gene determines which cell-reaction r is triggered if a defined morphogene m is measured with a concentration $k(m,z,t)$ higher than c_{min} and lower than c_{max} at the position of the cell. The measured concentration $k(m,z,t)$ of the morphogene m in the cell z can be modified by the receptivity-value $v(m,z,t)$ (see Eq. (4.19)), which is an internal value of the cell z, see Sect. 4.3.7. The genome does not change during the embryogenetic process. All cells share the same genome.

$$\Gamma = \{G_1, \ldots, G_N\} \tag{4.17}$$

$$G_i = (m, c_{\min}, c_{\max}, r), \quad i \in \{1, \ldots, N\} \tag{4.18}$$

$$k(m,z,t) = c(m,x,y,t) - v(m,z,t) \tag{4.19}$$

A cell in our model can react to a given morphogene concentration in different ways, e.g., emission of another morphogene, cell duplication and cell death. They are described in detail in Table 4.3.5. The ability of the "genes" in our model to get triggered by morphogenes is comparable to the mechanisms of gene expression and protein synthesis found in nature, e.g., in concepts of second-messenger mechanisms (Gomperts *et al.*, 2002) or transcription-coregulator mechanisms found in biological cells (Näär *et al.*, 2001). To achieve the required computational speed we simplified these processes for our concept of virtual embryogenesis. Further details about the genome and the interactions of genes can be found in Sects. 4.3.9 and 4.3.10.

Table 4.3 Possible reactions of a virtual cell.

Cell reaction	Description
Production of morphogenes	A cell emits a morphogene into the embryo, which diffuses throughout the embryo. The interaction of different morphogenes leads to the self-organisedprocess, mentioned in Sect. 4.3.3.
Cell duplication	The cell duplicates, which leads to a change of the embryo, due to virtual physics (mentioned in Sect. 4.3.6). The new shape can influence the diffusion process of morphogenes.
Cell death	The cell dies, which leads to a change of the embryo, due to virtual physics. For details see Sect. 4.3.6. Due to cell death, structures that where important during an early growth process, can be eliminated.
Changes in responsiveness	Changes the cell's responsiveness towards a morphogene. By changing these values the cell is able to differentiate. In the beginning of the virtual embryogenetic process cells are receptive to all morphogenes.
Changes of internal values	Internal values represent the predisposition for certain functions in the final neural network e.g.: the belonging to a defined subnet (as mentioned in Sect. 4.3.1)
Linking to neighbours	Builds a neural connection (dendrite) to a neighbouring cell. The weights between the neural cells are not touched by the embryogenesis.

4.3.6 Simulated Physics

In case of cell duplication or cell death the morphometrical structure of the embryo has to be reorganised. This process was implemented by assuming that cells interact physically with each other via pushing. Other complex interactions (e.g., cellular cohesion or adhesion) were not simulated. A duplicating cell determines the numbers of cells in the directions up, down, left and right. The cells in continuous rows are counted (until a gap is found). The whole continuous row of cells is shifted by one position in the respective direction where the number of cells is the smallest. The new cell is then placed on the new free position next to the duplicating cell. This way, the movement of cells during the growth process is simulated. Analogously to cell duplication, in case of a cell death the free patch of the dead cell is filled by shifting the whole row of cells towards the empty space. In both cases, cell death and cell duplication, always the smallest possible number of cells is moved. This way we simulate the physical situation in groups of cells without any physical connection like adheasion, where the physical inertia of subgroups of cells determines which cells move.

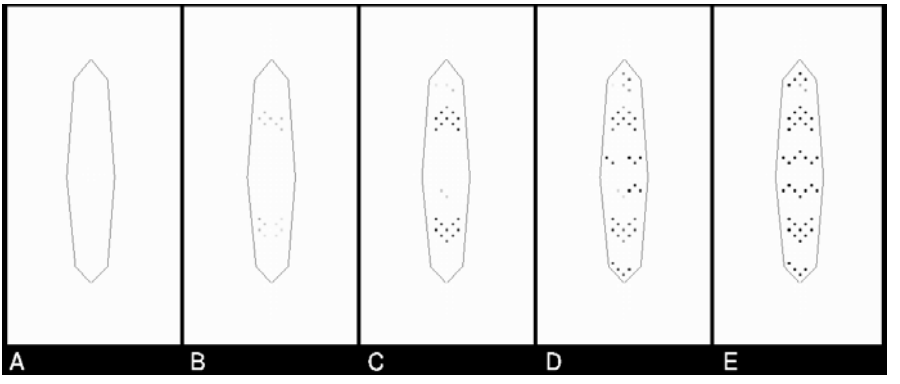


Fig. 4.20 Screen-shot of cell specialisation process in a virtual embryo. Specialised cells have a high value of a given internal status variable (indicated as grey dots, non-specialised cells are not drawn). Lines indicate the boundaries of the virtual embryo. A: Starting condition, B: Status of the embryo after 25 time steps. C: Status after 30 time steps. D: Status after 40 time steps. E: Final status of the embryo.

In our model some morphogenes can induce growth, other morphogenes can reduce growth. The interplay between these two groups of morphogenes during the embryogenetic process determines the size and the shape of the embryo. A big variety of shapes can emerge from this system, due to the different locations and the different time phases during embryogenesis when growth factors can be emitted.

4.3.7 Cell Specialisation

Morphogenes do not only influence the growth of the embryo by regulating cell duplication and cell death. They can also change internal status variables of cells (see Fig. 4.20). These values code for the receptivity for other morphogenes, the probability of building neural connections to other cells (mentioned below), or for properties that are necessary for the function of the resulting $\varepsilon\varepsilon$B (e.g., whether a cell belongs to a teaching or a controlling net, as mentioned in Sect. 4.3.1). Usually, the processes of cell specialisation take longer than the development of the shape of the embryo during embryogenesis, due to the fast response of cells that induce growth. Especially the shape of the embryo has a big influence on the interactions of different morphogenes, which leads to the specialisation of groups of cells. This corresponds to results found in nature (Müller, 2007). Some of the emerging processes can be interpreted as being "Turing processes" (Turing, 1952). Further details about cell specialisation can be found in Sect. 4.3.11.

4.3.8 Linkage

During the simulated embryogenesis, cells can build neural connections to other cells, see Fig. 4.21. The probability of a cell to build these connections, as well as

the distance to the linked cells, depends on the interplay between the morphogenes and the genome, see Fig. 4.22. The linkage process is a genetically defined action, as described in Sect. 4.3.5. In this way the degree of connectivity within a certain area of the embryo is determined by the embryological process. A cell remains linked, even if the cell is moved after linking. This can lead to long-distance connections throughout the whole embryo and to a structuring of the resulting $\varepsilon\varepsilon$B, see Fig. 4.23. If a cell is deleted during the embryological process (due to cell death) its links are also deleted. Not all cells within the embryo have to link to other cells. Although the virtual embryo has the purpose to shape a $\varepsilon\varepsilon$B, unlinked cells are not without function. They operate as morphological structuring cells in our model: these cells are needed for shaping the embryo due to growth or dying, as well as for shaping the gradients of morphogenes by functioning as spacer between areas of morphogene production and morphogene reception.

To investigate the outcome of the virtual embryologic process, the modelled embryo is allowed to grow and differentiate, until all growth and cell-differentiation is finished. That means that no more cell duplication events, cell death events, or cell linking events occurred over a long timespan. Also the distribution of growth factors within the embryo has to stay stable for the same timespan. If an embryo reaches this stable point of a complex equilibrium of development, it is defined as "finished". If the growth processes are not regulated well by the genome, the embryo can grow infinitely or vanish, due to triggering of cell death in all cells. To deal with "pathological" forms of embryos growing infinitely in our simulation, the embryological process is stopped if the number of cells reaches a certain bound. Resulting embryos with infinite growth or no stable result are rejected from further analysis.

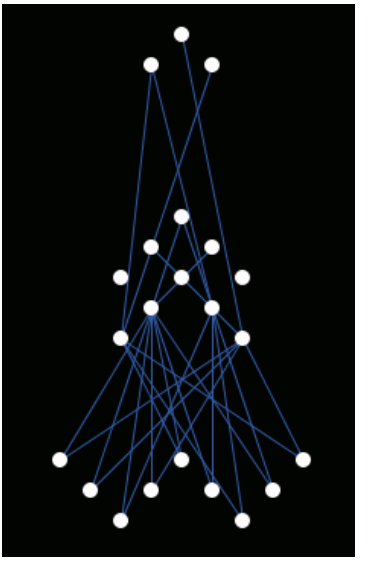


Fig. 4.21 Neural links between cells. For depicting reasons inter-cellular neural links (indicated by lines) are drawn in the embryo. The area of linked cells is zoomed. Cells are indicated by white circles. Reprinted from (Thenius *et al.*, 2009).

After the embryogenesis is finished successfully, the embryo is analysed and optimised, see Fig. 4.24. The network topology of the $\varepsilon\varepsilon$B is transferred into a structure that is readable for a neural network interpreter. Cells that had only a morphological structuring function during the embryogenesis and did not link during the embryogenetic process (as mentioned above) are excluded from the translation process to save computational time. These cells have no more influence on the shape or function of the ANN after the embryogenesis has finished. For details see Sect. 4.3.13.

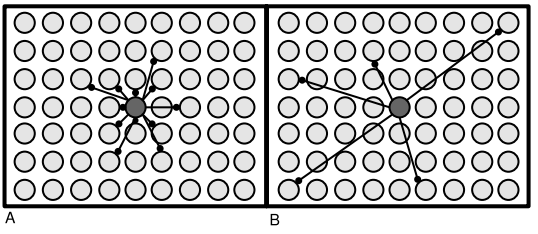


Fig. 4.22 Scheme of the linking process: The degree of connectivity and the distance of cells selected for connection is determined by the genome, by the morphogene level and by the internal state of the cell (e.g., the receptivity for a morphogene). A: A focal cell links with its closest neighbours with a high probability, resulting in a high density of local connections. B: Another cell has a low probability to build long distance connections, what results in a few long distance connections with cells further away. Reprinted from (Thenius *et al.*, 2009).

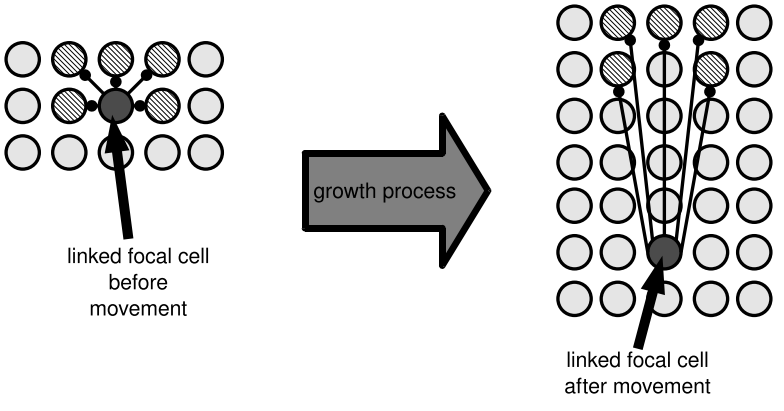


Fig. 4.23 Movement of linked cells. Once the focal cell is linked, the connections to other cells persist for the rest of the embryological process (except the focal cell dies), even if the cell is moved within the embryo. Reprinted from (Thenius *et al.*, 2009).

4.3.9 Depicting Genetic Structures and Feedbacks

While investigating the developing embryos we discovered several levels of complexity. Please notice, that every embryo in our model needs an initial external signal in form of a local morphogene gradient. In a first step, this gradient is placed in the centre of the world, directly below the first cell of the embryo. In the following, this external gradient is called the "prime gradient". It represents an external morphogene, influencing the embryo, e.g., gravity, light, temperature, morphogenes emitted by a maternal tissue.

The genome of the embryos is represented by a two-dimensional array, as described in Sect. 4.3.5. For better understanding we translated the genome into a graphic chart with our "Genome Logic Analyzer", see Fig. 4.25. In this graphic

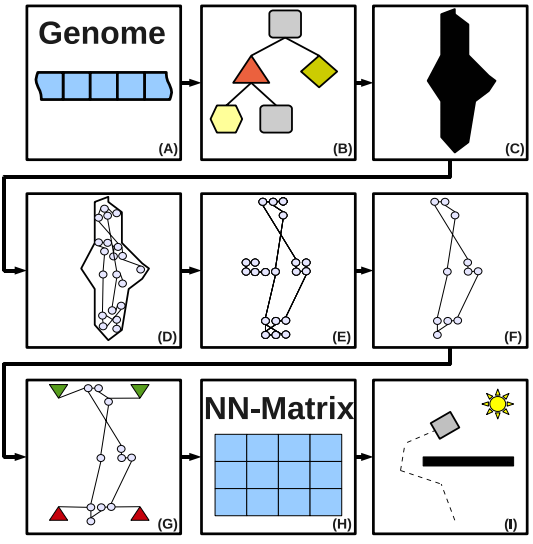


Fig. 4.24 Concept of shaping Artificial Neural Networks by virtual embryology. A genome (sub-figure A) that is able to perform complex operations including operations regarding the interpretation of the genome (sub-figure B), codes for the building process of a virtual embryo (sub-figure C). Some cells of this virtual embryo are able to develop links to other cells in the embryo and become neural cells this way (sub-figure D). After the embryogenetic process is finished, the Artificial Neural Network is extracted from the virtual embryo (sub-figure E). "Dead-end" connections within the network (that might develop during embryological process) are removed for the sake of calculation speed (sub-figure F). The Artificial Neural Network is then linked to sensor-interfaces and actuator-interfaces according to the definitions of the robot the network has to work in (sub-figure G). After this step, the network is translated into a data structure, that can be parsed by and Artificial Neural Network interpreter (sub-figure H) and tested in the robot or the simulation environment (sub-figure I).

chart, boxes indicate genes that lead to the emission of a morphogene, e.g. morphogene - coding gene "m1". Solid arrows indicate the triggering morphogene. Numbers next to solid arrows indicate the threshold for the triggering of a gene by a morphogene (for a detailed description of the triggering process and the genome please see Sect. 4.3.5). Other symbols indicate genes that lead to the production of proteins that influence the growth of the embryo. Upright hexagons (e.g., protein-coding gene "p102") indicate genes that produce proteins that lead to vertical growth, horizontal hexagons (e.g., protein-coding gene "p136") indicate genes that lead to horizontal growth. Triangles indicate genes that produce proteins (e.g., protein-coding genes "p13" and "p31") that increase or decrease the receptivity of a cell for a defined morphogene. Triangles pointing upwards (e.g., protein-coding gene "p13") indicate proteins that increase the receptivity of a cell, triangles pointing downward (e.g., protein-coding gene "p31") indicate proteins that decrease the receptivity of a cell. Genes that produce proteins that lead to the decreasing of

Table 4.4 Symbols used in the graphical chart of the genome

Symbol	Function of the gene
box	emission of morphogene
upright hexagon	vertical growth
horizontal hexagons	horizontal growth
diamond	omnidirectional growth
triangles pointing upwards	increase of receptivity for a morphogene
triangles pointing downwards	decrease of receptivity for a morphogene
pentagon pointing upwards	increase of internal value
pentagon pointing downwards	decrease of internal value
circle	linkage between cells

internal values are indicated by pentagons (e.g., protein-coding genes "p51" and "p68"). Pentagons pointing (upwards, e.g., protein-coding gene "p51") indicate proteins which increase an internal value of a cell. Pentagons pointing downwards (e.g., protein-coding gene "p68") indicate proteins which decrease an internal value of a cell. Genes that lead to the production of protein, that control the linking of cells are indicated by circles (e.g., protein-coding gene "p155"). Dashed lines indicate influences (and feedbacks) of proteins on morphogenes. The algebraic signs beside dashed lines indicate positive or negative feedback. All symbols which are used in the graphical charts are listed in Table 4.4.

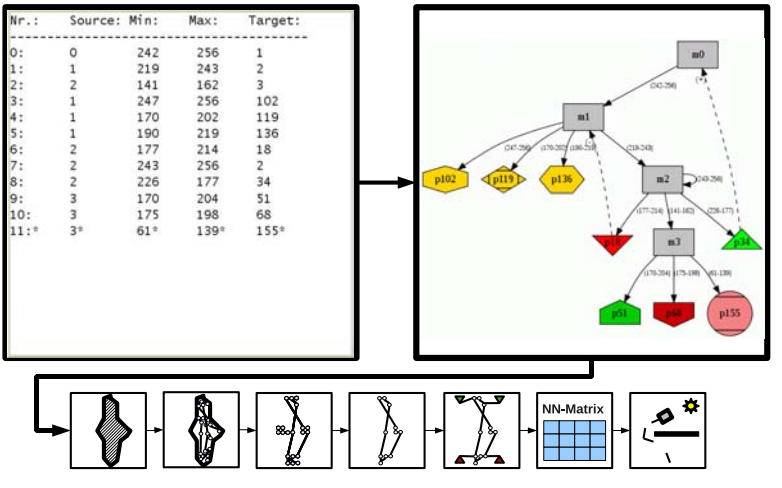


Fig. 4.25 "Genome Logic Analyzer": With this tool we are able to make graphical charts of the genome. The symbols are described in detail in Table 4.4.

4.3.10 Stable Growth due to Feedbacks in Genetic Structure

To solve the problem of cancerous (=infinite) growth in virtual embryos, mentioned in Sect. 4.3.8, the embryo has to evolve the ability to limit the growth process. This is possible through genetic structures that stop the emission of growth inducing morphogenes, or by lowering the effectiveness of growth inducing morphogenes. In the example depicted here, see Fig. 4.26, the lowering of the effectiveness of a morphogene is shown.

Fig. 4.26 shows a growth controlling genetic structure that is started by the morphogene producing gene "m2". This morphogene leads to its own emission (positive feedback), as well as the emission of the protein "p13". This protein decreases the receptivity of cells for the morphogene "m1", that induces the production of the growth inducing protein "p119" (negative feedback). The growth of the embryo depicted in Fig. 4.26 is limited. The growth process of the embryo is depicted in Fig. 4.27.

First, the embryo starts to grow. After 10 time steps, the speed of growth is decreasing. The reason for this slower growth is a decrease in cell receptivity for the growth inducing protein "p119" due to protein "p13". The production of the protein "p13" is induced by the genetic substructure beginning with morphogene "m1" which leads to a spreading of the morphogene "m2" throughout the whole embryo

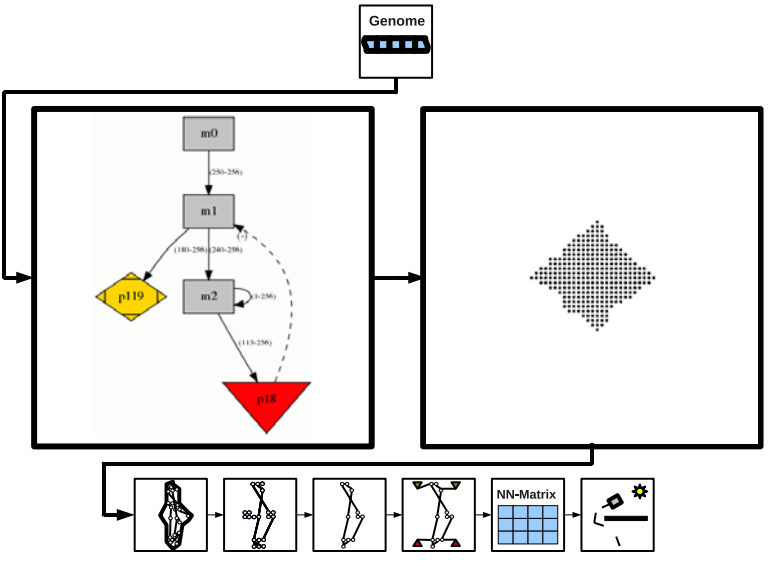


Fig. 4.26 Virtual embryo with limited growth: Right: The embryogenetic process results in a "stable" embryo, that stops growing. Cells are indicated by black dots. Left: Genetic Structure. The reason for the ability to stop growing lies in the genetic structure associated with morphogene "m2". It leads to the emission of protein "p13", which lowers the cells receptivity for morphogene "m1", that induces the emission of growth inducing protein "p119". The feedback, that limits the growth of the embryo, is depicted by a dashed line.

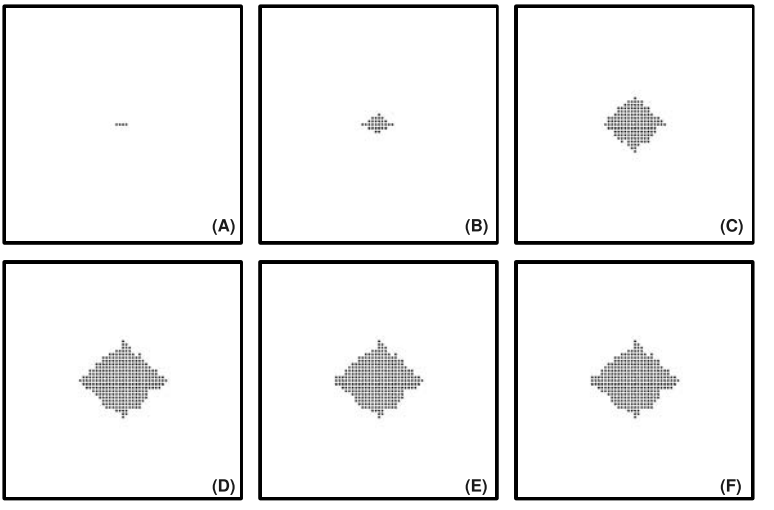


Fig. 4.27 Growth of a virtual embryo with limited growth: After a starting phase of about 9 time steps the growth of the embryo is reduced. After this point in time, the shape of the embryo stays the same, no more growth processes occur. A: time step 3; B: time step 6; C: time step 9; D: time step 12; E: time step 15; F: time step 18; Cells are indicated by black dots. Explanations of the symbols in the depiction of the genome can be found in Sects. 4.3.3 and 4.3.9.

as soon as the growth-inducing morphogene "m1" falls below a certain level in the far outer parts of the growing embryo.

4.3.11 Developing Complex Shapes

The combination of different genetic structures allows the design of complex shapes. In Figs. 4.28 and 4.29, two examples of embryos of more complex shape are shown.

As described in Sect. 4.3.7, the cells of the virtual embryo are able to differentiate into different kinds of tissues, including neural cells. The differentiation of cells is not only important for the coordination of growth, but also for structuring the embryo and the resulting $\varepsilon\varepsilon$B controller. Especially if different learning mechanisms, learning speeds or structured self-learning architectures (like in (Nolfi & Parisi, 1993)) should be used in a neural network (as mentioned in Sect. 4.3.1), it is necessary to mark areas within the network to behave differently during runtime. The process of cell-differentiation during embryogenesis is comparable to the process of branching the embryo during the growth process: under defined conditions (which are coded in the genome) very local combinations of morphogene levels trigger a defined protein, which leads to cell differentiation. The belonging of a cell to a certain tissue is defined by a group of internal variables that are modified by a cell-differentiating protein. In Fig. 4.30, an embryo with

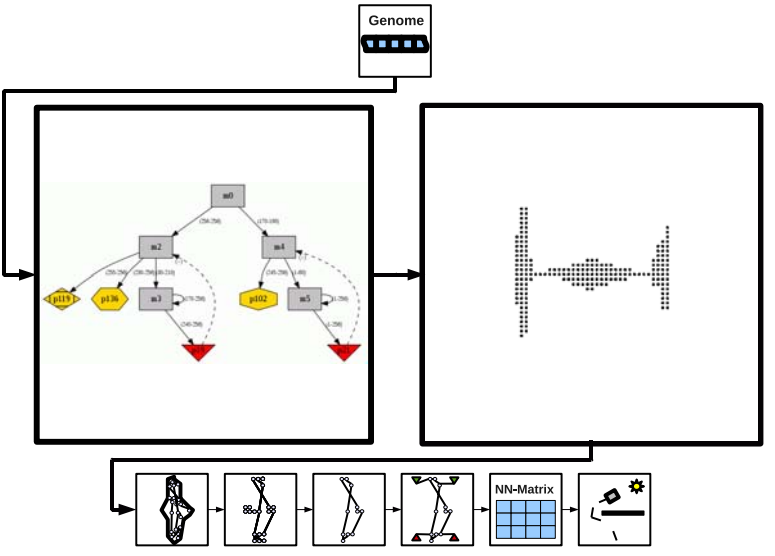


Fig. 4.28 Example for a complex shaped embryo. In the genome of the virtual embryo (left) we can see 2 main branches, one beginning with the morphogene-coding gene "m2" and one beginning with morphogene-coding gene "m4". The "m2" branch induces and regulates the horizontal growth of the embryo and is responsible for the development of the central part during embryogenesis (right: final shape of the embryo). The "m4" branch of the genome codes for the two vertical structures placed beside the embryo. Cells are indicated by black dots. Explanations of the symbols in the depiction of the genome can be found in Sects. 4.3.3 and 4.3.9.

differentiated cells is shown. In this simple example the differentiated cells are arranged to layers within the embryo.

4.3.12 The Growth of Neurons

One of the most important steps on the way from a genome to a $\varepsilon\varepsilon$B via virtual embryogenesis is the definition and linkage of neural cells within the virtual embryo. As described in Sect. 4.3.8, defined genes can trigger the growth of neural links of a cell. In Fig. 4.31 a simple multi-layered neural structure is shown. Due to locally differing concentrations of morphogenes it is possible, that sub-networks grow within differing parts of the embryo. Due to the possibility to modify internal values of the neural cells (as mentioned in Sect. 4.3.11), these sub-networks can differ, e.g., in their pre-disposition for leaning or in their function. One of the main advantages of the method described in this section is the possibility to structure neural networks this way via a genome which can be modified by artificial evolution (see also Sect. 4.3.1).

The genetic structure shown in Fig. 4.31 is comparable to the genetic structure of the embryo depicted in Fig. 4.30, but instead of decreasing or increasing of internal

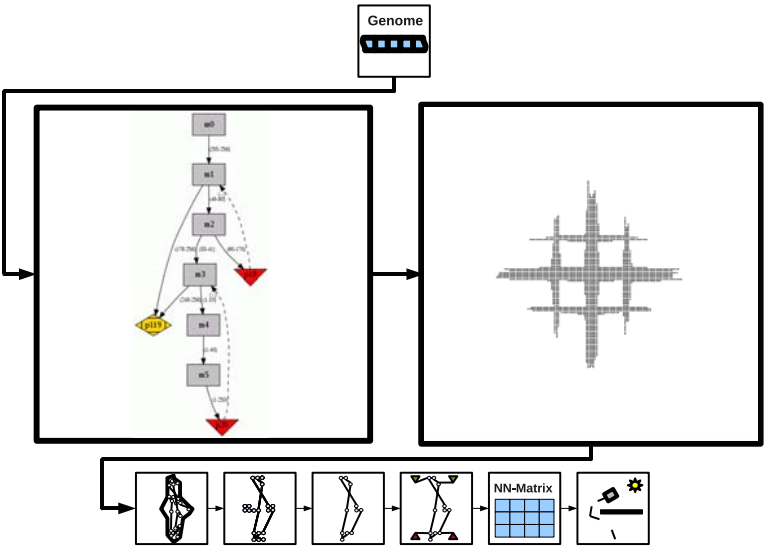


Fig. 4.29 Example for a complex shaped embryo. By using long, linear genetic structures (left), time delays between triggering and emission of morphogenes can be coded in the genome. These time delays determine the positioning of branches in the shape of the embryo (right). Cells are indicated by black dots. Explanations of the symbols in the depiction of the genome can be found in Sects. 4.3.3 and 4.3.9.

values, genes ("p165" and "p185") code for proteins that lead to a linkage of neural cells. The protein, coded by the gene "p165", is part of the genetic substructure (beginning with the morphogene coding gene "m2") that controls the vertical growth of the embryo. Due to its position, this gene induces the building of "long distance connections" all over the embryo. This process already takes place very early in the growth process of the virtual embryo, so that linked cells move in different directions and the neural connections reach throughout the whole embryo. The protein resulting from the triggering of the gene "p185" leads to the development of the spatially concentrated subnetworks. The subnetworks with high density are depicted by white lines in the right sub-figure of Fig. 4.31.

4.3.13 Translation

After the virtual embryo has stopped growing, it is necessary to extract the $\varepsilon\varepsilon$B from the embryo and to translate it into a structure that is readable for a neural network interpreter (as described in Sect. 4.3.8). In a first step, the network gets consolidated, see Fig. 4.24. This means, all neural cells that have no input or output are removed from the network. This "consolidation process" is an optimisation step which prevents unnecessary structures (like "dead end" connections). In a second step, the $\varepsilon\varepsilon$B has to be linked to the inputs (sensors) and outputs (actuators) of the

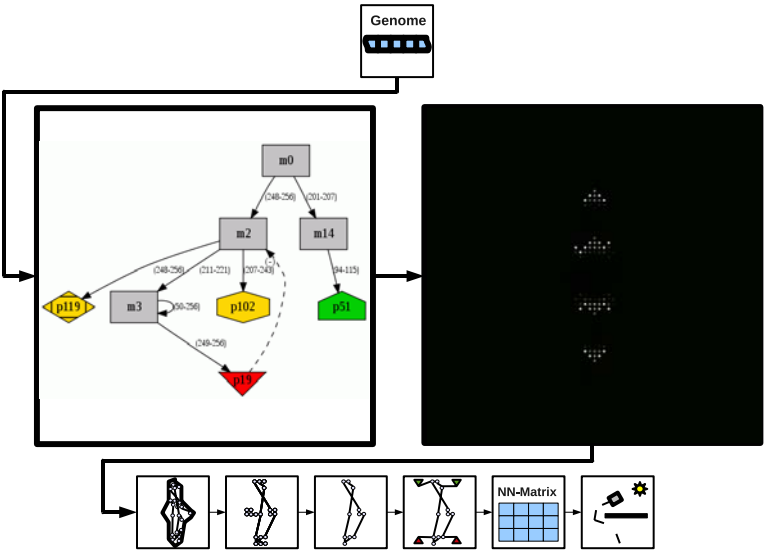


Fig. 4.30 Cell differentiation leads to structures within a virtual embryo. Left: The protein, expressed by the gene "p51", leads to a modification (increase) of a defined internal value. Right: The position of cells with a high internal value within the embryo is indicated by coloured dots. Cells with a low internal value are not depicted. The differentiated cells form stripes within the embryo. These cells can perform special functions within the final neural network.

robot, controlled by the network. The cells that become connected to the inputs and outputs are selected by their relative position in the virtual embryo corresponding to the position of sensors and actuators on the robot. In a third step the finalised neural network is translated into a structure of one-dimensional and two-dimensional arrays that can be parsed by a standard neural network interpreter. After this step the embryogenetically shaped $\varepsilon\varepsilon$B is ready for upload to the robot. The finalised neural network is uploaded to the robot. The neural network interpreter running on the robot parses the networks and operates on the network using adequate learning mechanisms.

4.3.14 Usability of Virtual Embryogenesis in the Context of Artificial Evolution for Shaping Artificial Neural Networks and Robot Controllers

One important ability of the described system, on which an artificial evolutionary process works, is the stability against lethal mutations, along with the ability to forward changes from the genetic level to the phenotypic (morphological) level. As shown in Fig. 4.32, (slightly) different embryological shapes can emerge from one "ancestor" by applying small changes to the genome. Please notice that these new

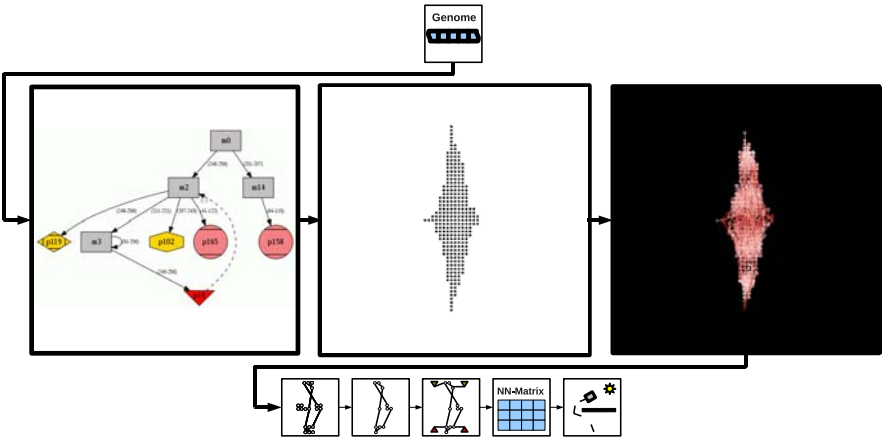


Fig. 4.31 Growth of an Artificial Neural Network in a virtual embryo. Right sub-figure: Screenshot of the neural network in the embryo. White lines indicate "short distance connections" within local subnetworks, red lines indicate "long distance connections" that reach throughout the whole embryo and connect the subnetworks with each other. Middle sub-figure: Shape of the embryo, neural connection not drawn. Left sub-figure: The genome defining the number and location of neural subnetworks in the embryo via self organisation processes (for details please see section 4.3.9).

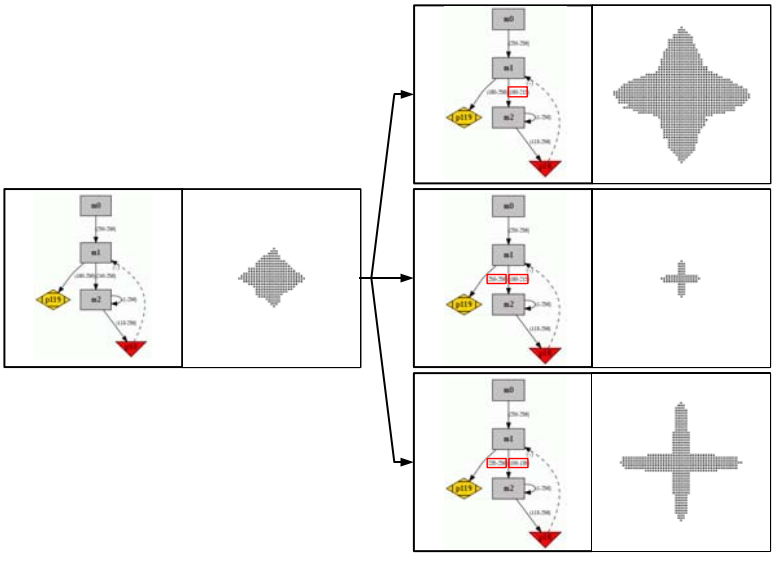


Fig. 4.32 Variations in the genome lead to variations in shape: If the genome of one "ancestor" (left sub-figure) is changed slightly, resulting embryos differ slightly in shape (right sub-figures). Changes in the genome are marked with red boxes. Reprinted from (Thenius *et al.*, 2009).

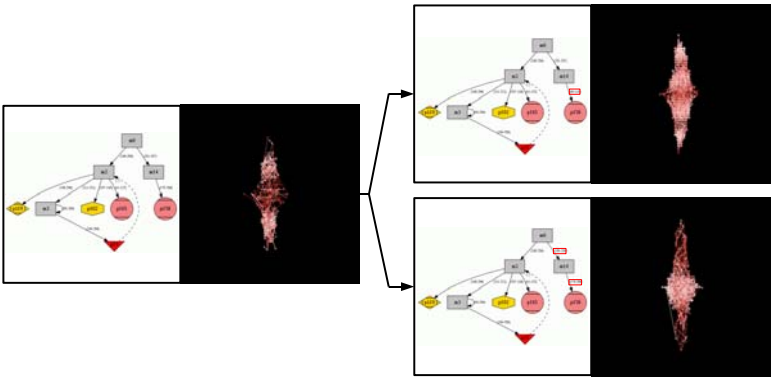


Fig. 4.33 Variations in the genome lead to variations in the shape of the resulting Artificial Neural Network: If the genome of on "ancestor" (left sub-figure, network with 2 layers) is changed, the resulting networks also change (right sub-figures, networks with 1, respectively 4 layers). Changes in the genome are marked with red boxes. Reprinted form (Thenius *et al.*, 2009).

shapes do not look completely different from its ancestor, but differ only slightly. These changes do not only take place on the level of shape, but also on the level of resulting $\varepsilon\varepsilon$B. As shown in Fig. 4.33, changes in the genome lead also to changes in the structure of the resulting network (i.e., the robot controller). This way the parameters of the $\varepsilon\varepsilon$B can change, e.g., the the number of layers, the number of cells within a layer, the micro-structure of a layer.

4.3.15 Subsumption of Section

Using the described approach of virtual embryogenesis we can simulate the development of an embryo from a simple hand-coded genome for the purpose of structuring an $\varepsilon\varepsilon$B, see Fig. 4.34. The final shape of the embryo, the connectedness of the embryo's cells, as well as the internal specialisation of cells (Fig. 4.34 D) are controlled by a system of feedbacks (Sect. 4.3.10) and delays, which arise due to the spreading speed of morphogenes (for details of morphogene implementation see Sect. 4.3.4). These feedbacks arise from the rule set described above, from the genome, from the spatial distribution of the cells within the embryo (see Fig. 4.34 A) and from the diffusion abilities of the morphogenes (see Fig. 4.34 B, C). The specialisation of cells within the embryo allows the development of different tissues, neural cells or structure cells, which have no neural but morphological function. The resulting patterns found in simulations of our model are comparable to patterns found in nature during embryological development. In Fig. 4.34 we compare the self-organised segmentation processes in our virtual embryo (Fig. 4.34A-D) to images from natural embryogenesis in *Drosophila m.* (Fig. 4.34 E). Similar segmentation patterns are also described by Kalthoff (Kalthoff, 1978).

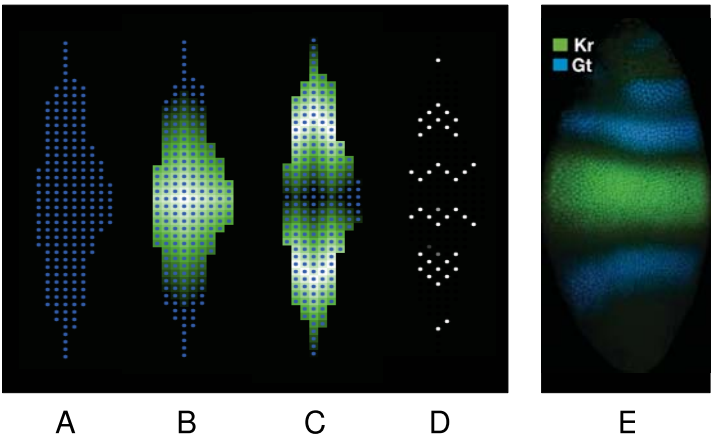


Fig. 4.34 Comparison of virtual embryogenesis in our model and real-world embryogenesis: A: Virtual embryo, consisting of cells (dots); B: Morphogene gradient in embryo; C: Gradient of another morphogene, inducing cell differentiation. D: Embryo consisting of differentiated cells (white dots) and non-differentiated cells (invisible) ; E: Natural example of gene expression: Activity domains of gap genes in larva (lateral view) of *Drosophila m.* (from (Jaeger *et al.*, 2004); 'Kr' and 'Gt' indicate gap genes.). Reprinted from (Thenius *et al.*, 2009).

For the described approach of virtual embryogenesis we used ideas from evolutionary developmental biology (EvoDevo). This approach produces results that are comparable to the products of natural developmental processes. The virtual embryogenetic processes described in this section have the potential to structure groups of cells on the level of body shape as well as at the level of micro-structure.

It is planned to combine the presented virtual embryogenesis with artificial evolution. The $\varepsilon\varepsilon$B that develops during the embryological process will be tested in a simulation environment. The fitness of a genome will be determined by the quality (e.g., learning ability) of the resulting "grown" $\varepsilon\varepsilon$B . Using this methodology novel and efficient ANN-structures are planned to be evolved, see Fig. 4.19. Additionally, much can be learned about the properties of basic processes that act during the biological evolution of brain structures (e.g., evolution of hierarchical brain-structures).

4.4 An Artificial Immune System for Robot Organisms

Jon Timmis, Andrew Tyrrell, Maizura Mokhtar,
Amelia Ritahani Ismail, Nick Owens, Ran Bi

Artificial Immune Systems (AIS) is a diverse area of research that attempts to bridge the divide between immunology and engineering and is developed through

the application of techniques such as mathematical and computational modeling of immunology, abstraction from those models into algorithm (and system) design and implementation in the context of engineering (Timmis *et al.*, 2008). AIS has become known as an area of computer science and engineering that uses immune system metaphors for the creation of novel solutions to problems (de Castro & Timmis, 2002).

The natural immune system consists of millions of immune cells performing various functions throughout the body, some identify various perturbations in the body's internal environment and produce appropriate response(s) to eliminate or reduce the effect of the perturbation. This perturbation can be the affect of an invasion from infectious body(s), cellular damage, unusual cellular behaviour (cancerous cells) or unusual chemical in-balance in the body. The immune system provides a response by balancing the degree of perturbation, the strength of the detected infection and the type and severity of the immune response. This balance is to prevent the immune system from attacking its own-self that could create an auto-immune response. This balance ensures that the homeostasis of the body is maintained despite perturbation occurring to the body. We argue that these functions provided by the immune system are essential for the creation of homeostatic operation of robotic systems, where long term autonomy is required and tolerance to errors is essential.

Robotic swarms and robotic organisms, the subject of this book, falls within the context of collective robotics system. These systems can be defined as a society of robots that coordinates its behaviour via interaction and cooperation with other robots in the society. This allows for the robotics system to achieve a collective goal. Two important issues that need to be addressed for these system are the assurance that (1) the stability of the collective system is maintained and (2) each robot in the collective provides useful information to afford stability. These two issues are important as the robots are constantly exposed to changes in the environment. Such changes can alter the states of the robot itself, and the entire organism. Therefore, it is necessary to regulate and modulate such changes so that the changes will not significantly affect the system behaviour in a detrimental way such as to effect the accomplishment of the system goal.

This chapter outlines a method, that we argue, is able to provide *artificial immunity* to both swarm and organism level robotic systems. Such artificial immunity affords homeostasis at both the individual and collective level by means of fault tolerance in a variety of guises. The idea of immune inspired homeostasis is not new, indeed, the authors have proposed an outline of such an approach in (Owens *et al.*, 2007).We build on those ideas and propose a vision of an immune-inspired framework for homeostasis in the context of modular robotic systems.

4.4.1 A Biological and Engineering Perspective

4.4.1.1 Robustness, Self-organisation and Adaptation

Robustness, self-organisation and adaptation are key properties that have been a source of inspiration for research in swarm robotic systems. From a biological

perspective, robustness is a fundamental characteristic (Kitano, 2007). In biological systems robustness is defined by (Kitano *et al.*, 2004) as:

> "robustness is a property that allows a system to maintain its functions despite external and internal perturbations. It is one of the fundamental and ubiquitously observed systems-level phenomena that cannot be understood by looking at the individual components. A system must be robust to function in unpredictable environments using unreliable components."

Self-organisation, or decentralised control, is widespread in biological systems, including cells, organisms, and groups that possesses a large number of subunits, and these subunits lack either the communicational abilities or the computational abilities, or both, that are needed to implement centralised control (Camazine *et al.*, 2001). Self-organisation is defined by (Camazine *et al.*, 2001) as:

> "a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. The rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern."

Adaptation is a basic phenomena in biology (Williams, 1966), whereby an organism becomes better suited to its habitat. The term also may refer to the adaptation of the organism, which is especially important for an organism's survival. For example, the adaptation of horses' teeth to the grinding of grass, or their ability to run faster and escape from predators. Such adaptations are produced in a variable population by the better suited forms reproducing more successfully, that is, by natural selection (Williams, 1966). As discussed in (Williams, 1966), adaptive traits may be structural, behavioural or psychological. In addition, (Williams, 1966), mentions that structural adaptations are physical features of an organism such as shape, body covering, defensive or offensive armament. Behavioural adaptations are composed of inherited behaviour chains and/or the ability to learn: behaviours may be inherited in detail (instincts), or a tendency for learning may be inherited for example searching food, mating and vocalisation. Finally, physiological adaptations may permit the organism to perform for instance making venom or secrete slime (Williams, 1966). What is interesting to note is that there are multiple timescales of adaptation, ranging from short term to long term. This idea of multiple timescale adaptation, is something that we wish to exploit in our robotic system.

4.4.1.2 Swarm Robotics

It has been argued that the key benefit of swarm robotics approach is robustness, which manifests itself in a number of ways (Şahin & Winfield, 2008). Firstly, as

a swarm of robots consists of a number of relatively simple, and typically, homogeneous robots which are not pre-assigned to explicit roles or tasks within the swarm, the swarm should self-organise, or dynamically restructure, the way individual robots are arranged. Secondly, it is claimed that the swarm approach is highly tolerant to the failure of individual robots. The failure of an individual robots should, in principle, not affect the goal of the system as a whole. Thirdly, due to decentralised control in a swarm there is, in principle, no common-mode failure point or vulnerability in the swarm. Indeed, it could be said that the high level of robustness evident in robotic swarms comes for free in the sense that it is intrinsic to the swarm approach, which contrasts with the high engineering cost of fault tolerance in conventional robotic systems (Şahin & Winfield, 2008). However, this view is potentially problematic, as if the underlying behaviours of the swarm were not designed carefully, this could have major impact on the performance of the swarm, irrespective of the size. Furthermore, (Şahin & Winfield, 2008) highlight a number of challenging problems in swarm robotics systems that need attention, namely algorithm design, implementation and testing, and, analysis and modelling. In algorithm design, swarm roboticists face the problem of designing both the physical morphology and behaviours of the individual robots such that when those robots interact with each other and their environment, the desired overall collective behaviours will emerge. At present, there are no principled approaches to the design of low-level behaviours for a given desired collective behaviour. Secondly, in implementation and test: to build and rigorously test a swarm of robots in the laboratory requires a considerable experimental infrastructure. Real-robot experiments thus typically proceed hand-in-hand with simulation and good tools are essential. Thirdly, in analysis and modelling: a robotic swarm is typically a stochastic, non-linear system and constructing mathematical models for both validation and parameter optimisation is challenging, this again is a topic contained within this book, see Sect. 1.3.

Winfield *et al* (Winfield & Nembrini, 2006) claim in their paper that even though swarms exhibit high level of robustness, such claims are frequently not supported by empirical or theoretical analysis. Winfield *et al* also raise questions regarding robustness: such as what does robustness actually mean and how can one measure robustness or the fault tolerance of a robotic swarm. In answering those questions, (Winfield & Nembrini, 2006) explores fault tolerance in robot swarms through Failure Mode and Effect Analysis (FMEA)[1] illustrating by a case study of wireless connected robot swarm, in both simulation and real laboratory experiments.

There is a clear connection between these issues raised in the context of swarm robotics, and collective robotics (the subject of this book). Issues of fault tolerance are paramount in either application, and the ability of systems to detect errors, and recover from those errors is central to survivability of the organism or swarm.

[1] A failure modes and effects analysis (FMEA) is a procedure for analysis of potential failure modes within a system for classification by severity or determination of the effect of failures on the system. Failure modes are any errors or defects in a process, design, or item, leading to the studying the consequences of those failures to the systems.

4.4.1.3 Fault Tolerance and Dependable Swarms

From an engineering standpoint, the design of complex distributed systems based upon the swarm intelligence paradigm is compelling but problematical. Due to the characteristics of the system itself where no hierarchical command and control structure are needed and hence no common mode of failure point or vulnerability can be identified. Each individual agent makes decisions autonomously, based upon local sensing and communication (Bonabeau *et al.*, 1999). Systems with these characteristics could, potentially, exhibit very high levels of robustness, in the sense of tolerance to failure of individual agents and much higher levels of robustness (Winfield *et al.*, 2006b). Simple units can, in fact, collaborate in achieving their common goal without the need of being aware of the rest of the group. Resilience achieved in this way makes the paradigm very appealing in many applications; however one or more faulty robots may jeopardise the success of the overall mission. As noted by (Winfield *et al.*, 2006b), there are two reasons for undesirable behaviours in swarm robotics: *random errors,* or *systematic (design) errors* . Random errors are those due to hardware or component faults. The likelihood that random errors cause undesirable behaviours can be reduced, in the first instance, by employing high reliability components (Winfield *et al.*, 2006b). However, these systems also need to be fault tolerant, through redundancy for example. Systematic errors are those aspects of the design that will allow the system to exhibit undesirable behaviours. Analysis of systematic errors for the swarm as a whole is much more problematical, particularly if the desired behaviours are emergent (Winfield *et al.*, 2006b). (Winfield *et al.*, 2006b), proposed the idea of dependable swarm, which is described as:

> "complex distributed system, designed using the Swarm Intelligence paradigm, which meets standards of analysis, design and test that would give sufficient confidence that the system could be employed in critical applications" - *Alan Winfield*

As highlighted by (Winfield *et al.*, 2006b), in the swarm intelligence literature, robustness sometimes refers to simplicity and hence functional and mechanical reliability of simple, even minimalist robots that comprise a swarm. Sometimes, robustness also refers to the ability of the swarm to cope with a demanding operational environment (Mondada *et al.*, 2002) but most often robustness refers to the swarm's tolerance to the failure of one or more individual robots (Kazadi *et al.*, 2004). Thus, (Winfield & Nembrini, 2006) summarised that a robot swarm is robust if the swarm is:

- a completely distributed system and therefore has no common-mode failure point
- comprised of simple and hence functionally and mechanically reliable individual robots
- tolerant to noise and uncertainties in the operational environment

- tolerant to the failure of one or more robots without compromising the desired overall swarm behaviours
- tolerant to individual robots who fail in such a way as to thwart the overall desired swarm behaviour.

4.4.1.4 Artificial Immune Systems

Artificial Immune Systems (AIS) have been defined by (de Castro & Timmis, 2002) as:

> "adaptive systems, inspired by theoretical immunology and observed immune functions, principle and models, which are applied to problem solving."

The immune system is a complex system that undertakes a myriad of tasks. The abilities of the immune system have helped to inspire computer scientists to build systems that *mimic*, in some way, various properties of the immune system. This field of research, Artificial Immune Systems (AIS), has seen the application of immune inspired algorithms to problems such as robotic control, network intrusion detection, fault tolerance and machine learning (Hart & Timmis, 2008). From a computational point of view, the immune system has many desirable properties that might be emulated in computer systems. These properties are such things as robustness, adaptability, diversity, scalability, multiple interactions on a variety of timescales and so on. The main developments within AIS, have focussed on four main immunological theories: clonal selection, immune networks, negative selection and danger theory (Timmis *et al.*, 2008). Researchers in AIS have concentrated, for the most part, on the *learning* and *memory* mechanisms of the immune system inherent in clonal selection and immune networks, and the negative selection principle for the generation of *detectors* that are capable of classifying changes in *self*, and ideas from innate immunity, for correlating multiple signals over time to attempt to identify breaches in computer networks (Timmis *et al.*, 2008). In our work, we have chosen to focus on both the innate and adaptive aspects of the immune system, so we briefly outline some basic immune system terminology to prepare the reader for the rest of the chapter.

The vertebrate immune system (the one which has been used to inspire the vast majority of AIS to date) is composed of diverse sets of cells and molecules. These work in collaboration with other systems, such as the neural and endocrine, to maintain a steady state of operation within the host: this ability is termed *homeostasis*. The role of the immune system is typically viewed as one of protection from infectious agents such as viruses, bacteria, fungi and other parasites. On the surface of these agents are antigens that allow the identification of the invading agents (pathogens) by the immune cells and molecules, which in turn provoke an immune response. There are two basic types of immunity, innate and adaptive. Innate immunity is not directed towards specific pathogens, but against any pathogen that enter the body. The innate immune system plays a vital role in the initiation and

regulation of immune responses, including adaptive immune responses. Specialised cells of the innate immune system evolved so as to recognise and bind to common molecular patterns found only in microorganisms. However, the innate immune system is by no means a complete solution to protecting the body.

Adaptive, or acquired immunity, is directed against specific invaders, with adaptive immune cells being modified by exposure to such invaders. The adaptive immune system mainly consists of lymphocytes, which are white blood cells, more specifically B and T cells. These cells aid in the process of recognising and destroying specific substances. Any substance that is capable of generating such a response from the lymphocytes is called an antigen or immunogen. Antigens are not the invading microorganisms themselves; they are substances such as toxins or enzymes in the microorganisms that the immune system considers foreign. Adaptive immune responses are normally directed against the antigen that provoked them and are said to be antigen-specific.

The clonal selection theory (CST) (Burnet, 1959) is the theory used to explain the basic response of the adaptive immune system to an antigenic stimulus. It establishes the idea that only those cells capable of recognising an antigenic stimulus will proliferate, thus being selected against those that do not. Clonal selection operates on both T cells and B cells. In the case of B cells, when their antigen receptors (antibodies) bind with an antigen, the B cell becomes activated and begins to proliferate producing new B cell clones that are an exact copy of the parent B cell. The clones then undergo somatic hypermutation and produce antibodies that are specific to the invading antigen (Berek & Ziegner, 1993). After proliferation, B cells differentiate into *plasma cells* or long-lived B *memory cells*. Plasma cells produce large amounts of *antibodies* which will attach themselves to the antigen and act as a type of *tag* for other immune cells to pick up on and remove from the system. This whole process is known as *affinity maturation*.

Memory cells help the immune system to be protective over periods of time. In the normal course of the evolution of the immune system, an organism would be expected to encounter a given antigen repeatedly during its lifetime. The initial exposure to an antigen that stimulates an adaptive immune response is handled by a small number of B cells, each producing antibodies of different affinity. Storing some high affinity antibody producing cells (memory cells) from the first infection, so as to form a large initial specific B cell sub-population for subsequent encounters, considerably enhances the effectiveness of the immune response to secondary encounters. Such a strategy ensures that both the speed and accuracy of the immune response becomes successively stronger after each infection.

Danger theory attempts to explain the nature and workings of the immune response in a way different to the more traditional clonal selection view (Matzinger, 1997). Matzinger criticises this idea, as she states that observations demonstrate that it may sometimes be necessary for the body to attack itself and conversely the immune system may not attack cells it knows to be foreign (this is not possible under the classical clonal selection theory). Matzinger argues a more plausible way to describe the triggering of an immune response is a reaction to a stimulus the body considers harmful. This might be seen as a very small change but

in reality this is real shift in thinking about how the immune system responds to pathogens. In essence, this model allows for foreign and immune cells to exist together, a situation impossible in the traditional standpoint. When under attack, cells dying unnaturally may release a danger signal, that disperses to cover a small area around that cell: a danger area. It is within this and only within this area that the immune system becomes active and will concentrate its attack against any antigen within it. The complex interaction of B-cells, dendritic cells, and T-cells takes place in lymph nodes which are located throughout the body.

The immune system is complex, and there are many actors and processes within the immune system. For our work, we have decided to focus on a number of areas which can be combined to a unique fault tolerance mechanism that has the ability to detect many types of errors, over multiple time-scales, and also has the ability to initiate *repairing* mechanisms within the robotic collective.

4.4.1.5 Homeostasis and Fault Tolerance: Requirements for an Artificial Immune System

Fault tolerance is defined as the "ability to achieve dependability or reliance when the system is functioning with faults" (Timmis *et al.*, 2002). A fault is hypothesised as the cause of error. An error is part of the system that is liable to lead to subsequent failure. A failure occurs when a system service deviates from its expected behaviour. Fault tolerance consists of:

1. *Error processing:* aims to identify an error in the system before such error can cause failure. Error processing consists of error detection and error diagnosis that pin-points the cause of a fault.
2. *Fault recovery:* error recovery that provides methods of recovering the system to normal behaviour when an error is detected and processed.
3. *Fault treatment:* prevents faults from re-occurring in the system after its initial error detection.

Previous work in (Owens *et al.*, 2007) outlines the concept of immune homeostasis in the context of electronic systems and fault tolerance. The authors develop the concept of three desirable properties for homeostatic control systems, taken from (Owens *et al.*, 2007):

- Prediction. Vander (Vander *et al.*, 1990) determines this as feed-forward regulation. In response to an environmental change the homeostatic control system manipulates the internal environment in order to avoid a deviation from a set point before it has happened.
- Innate and Adaptive Response. The homeostatic control system is built up of innate and adaptive reflexes which are used to bring homeostatic variables back to set points. The innate reflexes are involuntary, unpremeditated and unlearned, and are instigated in response to a particular stimulus, internal or external. As one would imagine, adaptive reflexes are learned to correct unforeseen deviations from set points. Vander also states that all reflexes, innate or adaptive, are subject to further learning.

- Acclimatisation. Although encompassed by both adaptive responses and re- setting of set points, it is an important enough property in its own right. It represents the ability for a set point to semi-permanently change in response to semi-permanent change in the environment. To aid explanation we take the analogy in (Vander *et al.*, 1990) of a runner who is asked to run for 8 consecutive days in a hot room (a room hotter than the runner's normal environment). Details of the runner's sweating are recorded. By the 8th day the runner starts to sweat earlier and in far greater quantities than the 1st day, this allows to the runner to limit the deviation of the temperature homeostatic variable from its set point. The "sweating" homeostatic set point has acclimatised to the new environment. When the runner returns to running in the original environment the set point will, over a number of days, acclimatise back to the original.

In (Owens *et al.*, 2007), building on their early desirable properties for a homeostatic systems, the authors provide an outline of what such an artificial system needs to meet those properties. First, the control system must have the ability to *arbitrate*, by which is meant that given a certain error and operational state of the unit, the control system must be able to decide (arbitrate) between various possible responses. Second, the control system will be able to *correlate* between the sensor values and any homeostatic error. Third, the control system should be able to *learn* by experience and improve over time. Fourth, the control system should be capable of *prediction*, that is be able to identify ahead of time the consequence of an error and be able to recommend corrective action and finally, the system should have the ability to *acclimatise*, which is linked to the ability of the system to learn, in that should the normal operational environment change over time, then the control system should adapt to that change without raising an alarm of an error.

In this chapter, we do not propose a control system in the classic sense, but a system that sits between the world of the sensors (discussed in Sect. 3.2) and the behaviour based controller discussed in Sect. 4.1.

4.4.2 An Immune-inspired Architecture for Fault Tolerance in Swarm and Collective Robotic Systems

4.4.2.1 Overview of Architecture

Fig. 4.35 outlines, at a high level, our proposed framework. We split the framework into two separate levels: innate and adaptive, around the concept of an artificial lymph node. The natural immune system can be conceptually divided into two layers, known as the innate and adaptive immune system. The innate level is akin to a pre-programmed ability to identify specific patterns which then results in specific responses. The adaptive level is akin to a non-specific response which adapts over the life-time of the host, which in some cases may improve over time. It is envisaged that the AIS is encoded in a genome that is held within the robotic unit, and this itself will adapt over time. More information on the genome can be found in Chapter 5.

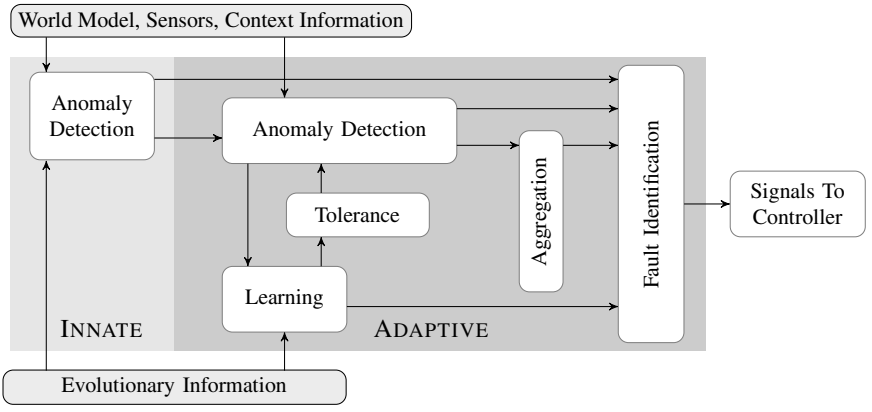


Fig. 4.35 A Lymph Node Framework to be instantiated on a single robotic unit. Input is derived from the world model, learnt context information, and genome information and passed to the innate layer. Evolutionary information is also passed to the adaptive layer to act as a reinforcement feedback during the learning process

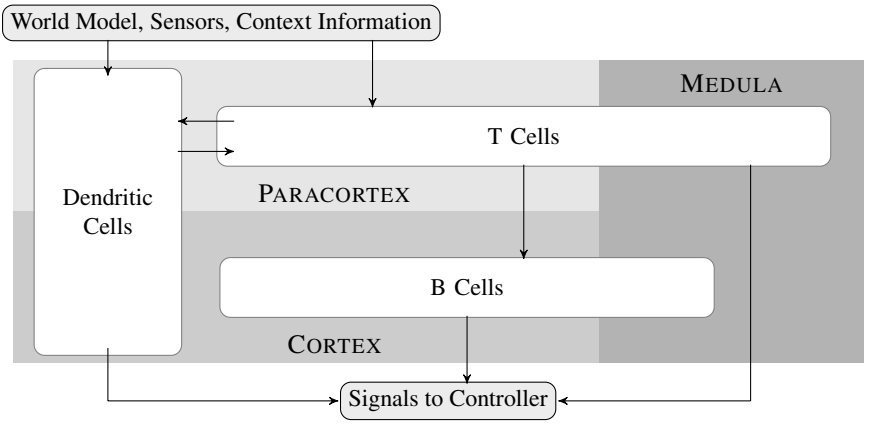


Fig. 4.36 Mapping the Lymph Node Framework to Functional Lymph Node Areas. Here input is the same as for Fig. 4.35, but we we show the receiving agents of the external signals, the dendritic cells, B-cells and T-cells (genome information only)

To clarify what is meant by anomaly detection we define the following three types of anomaly:

- *type 0*: There are some constraints on a system's operation for which we have *a priori* knowledge. An example could be the valid operating temperatures for a component in the system. We may build these constraints directly into our innate anomaly detection system. We determine a violation of one of these *a priori* constraints a *type 0* anomaly.

- *type 1*: We wish to evaluate the probability of readings from a set of sensors conditional on the history of those sensors. That is, are the new sensor readings "similar to" recent sensor readings. We detect a *type 1* anomaly if new sensor readings are sufficiently different from their history (e.g. below a threshold probability, or far away under an appropriate metric).
- *type 2*: The detection of a *type 1* anomaly relies on the assumption that the history of sensor readings provides a good model with which to evaluate new sensor readings. Consequently, we wish to test the validity of this assumption and we may do so by tracking macroscopic changes in the sensor history. When the macroscopic properties of the history of sensor readings are too large or occur too quickly we may be unable to perform *type 1* detection. Further, large and fast macroscopic changes can be an indicator of a large environmental change or that a relationship between multiple sensors has broken. As such, we detect a *type 2* anomaly when there is an appropriately large or fast change in a history of sensor readings.

Our innate anomaly detection system performs *type 0* anomaly detection and our adaptive anomaly detection system performs *type 1 & type 2* anomaly detection.

Examples of the three anomaly types can be given in terms of detecting the ambient light level on the outside of a building. The light sensor may have the operational constraint that its readings are only reliable in a restricted range of light levels, outside this range the sensor may be inaccurate. A *type 0* anomaly should be detected if the sensor provides a reading that falls outside the region of reliability. Next, if a new sensor reading reports a dark ambient light and all previous light readings report very bright conditions it is likely that the sensor is faulty so a *type 1* anomaly is detected. Then type *type 2* anomalies are concerned with tracking the changes in light from day to night. There is an expected rate at which the ambient light level will change, deviations from this rate could suggest a faulty sensor or an unexpected change in the environment and so a *type 2* anomaly. In this situation the rate of change of light levels which cause a *type 2* anomaly are slow enough that a *type 1* anomaly is never detected. That is, the changes are slow enough that every sensor reading is similar to the other readings in surrounding time window used for *type 1* detection.

Such detection can be used at two levels in the collective. First, it can be useful at the individual level, as identifying potential problems within a unit allows the AIS to inform the controller that certain sensors might not be reliable, so alternative strategies might be employed, or alternative data sources might be used as a consequence of the error. Importantly in the context of collective robotics, if we identify a malfunctioning robot before it joins the collective, a decision can be made disallowing the robot joining the collective: thus protecting the overall stability of the organism. Likewise, during organism function, if a faulty robot is identified, then it might be able to be removed from the organism in an attempt to preserve the overall functionality of the organism. We now explore how such a system might be implemented.

4.4.3 Innate Layer

Our innate component, Fig. 4.35, consists of a process of anomaly detection that takes data from internal and external sources. The innate system only adapts on an evolutionary timescale, that is directed by the genome. As such, the transfer of immunological genetic information between robotic units can be implemented. We make use of the innate immune system antigen presenting cell (APC), or Dendritic Cell (DC) analogy (Mokhtar *et al.*, 2008), see Fig. 4.36. This layer will allow for an effective "filtering" of the data for detecting potential anomalies in the data stream, in this case of *type 0* anomalies. The innate layer has the ability to *correlate* a variety of input sources over time and identifies the presence of an error based on the correlations of these signals. The innate layer also has the ability for prioritising an alarm, by indicating the severity of the danger and; *learning* by the incorporation of new knowledge over time on an evolutionary timescale via changing at the genome level. In the context of Fig. 4.35 we now describe, at a functional level, the operation of our proposed system.

At the innate level, which is different from a more traditional approach for anomaly detection, we do not need to define the normal operational state beforehand, but we do need to define *operational boundaries* of the components being monitored, hence *type 0* error detection. Operational boundaries of the components can be defined from the component's datasheet or the module's parameters. The innate AIS will perform this detection by correlating these parameters values over time, and attempt to detect the occurrence of errors based on these values. Therefore, faults and other environmental changes (anomaly) that can affect the performance of the robotic unit will be identified according to the changes in the operational activity of the unit over a certain period of time. Should such an anomaly be detected, a message is passed to the fault identification process (part of the adaptive AIS), and also to the adaptive learning process and is used to help make decisions on any remedial action that should be taken.

Our proposed approach, produces an output value (*DangerSig*) that provides a normalised collective indication that the weighted sum, $O_C(t)$ of the three signals (PAMP (P) , danger (D) and safe signals (S)) (Eq. 4.20) is greater (or lesser, depending on its application) than a threshold value within a certain time window. These three signals assimilate to how the biological dendritic cells discriminate self and non-self antigens. Dendritic cells differentiate to its full maturation state when it receives more signals corresponding to non-self antigen: the PAMP and danger signals than the signal corresponding to self antigen: the safe signal. PAMP signal is the signal produced when the innate immune cell detects PAMPs or Pathogen-Associated Molecules Patterns. Danger Signals are signals produced when cells die necrotically. Safe signals are signals produced because of cells apoptosis.

This weighted sum provides an indication of the state of the component or module at time t:

$$O_c(t) = \frac{w_p P(t) + w_s S(t) + w_d D(t)}{w_p + w_s + w_d} \tag{4.20}$$

We then apply a threshold to $O_c(t)$:

$$D_\tau(t) = \begin{cases} 0 & O_c(t) < \tau \\ 1 & O_c(t) \geq \tau \end{cases} \tag{4.21}$$

We can now describe the danger count over a time window $(t-\omega)$ to (t):

$$DangerSig = \frac{1}{\omega+1} \sum_{i=0}^{\omega} D_\tau(t-\omega) \tag{4.22}$$

We must define the PAMP, danger and safety signals.

1. PAMP *(P)* provides an indication of definite anomaly in the *module*.
   For example, if we are monitoring the power module of the unit: PAMP = 1 when the remaining charge capacity for the robot falls below a threshold. The threshold could be sensibly defined by statistics of the dissipation rate of the power module.
2. Danger signal *(D)* provides an indication of possible anomaly in the monitored input space of a *module*. If it is the case that certain sensors in the system should exhibit a slow rate of change then a fast rate of change in these sensors could be a good indication of danger. Giving an example relating to the power module, a high rate of change in the dissipation rate could be a good indication a faulty power module or component in the system and so a good choice for a danger signal.
3. Safe signal *(S)* provides an indication of normal behaviour (homeostasis) to the *module*. Safe signal is calculated based on the input values.

DangerSig is a normalised value in the range $[0,1]$ which indicates the number of times the value of O_C is less than a threshold value within a time window. If the value of *DangerSig* is 1, then this provides an indication, that during the time window, there is zero confidence of anomalous behaviour in the monitored module, i.e. everything is operating as it should be. If the value of *DangerSig* decreases, the confidence of anomaly present increases. If *DangerSig* is 0, therefore, there are definite anomalous behaviour occurring in the module.

4.4.4 Adaptive Layer

The adaptive artificial layer, Fig. 4.35, is analogue to the collaborative effort of B-cells and T-cells, key actors in the natural immune system's adaptive layer. Our system, therefore, employs two distinct and complimentary methods of identifying anomalies, one taking inspiration from the behaviour of B-cells, the other takes inspiration from the operation of T-Cells. The B-cell approach maintains a B-cell population and is akin to instance based learning approach where actual instances of data are stored in a feature vector and are used to compare against the current readings from the robot sensors. We employ a time window, similar to the approach taken in (de Lemos *et al.*, 2007) where we build up a population of detectors capable of identifying potential anomalies based on examples seen in the past and learnt over time.

The T-cell approach takes inspiration from the T-cell receptor and performs estimates of the densities associated with the distributions of sensor values. A *type 1* anomaly occurs when a new sensor reading has a probability below a threshold with respect to the distribution estimate. The *type 2* anomaly is then defined through statistics on this probability distribution estimate, such as the rate of change of the distribution.

Whilst the two approaches are inspired by complimenting and interacting immunological concepts they are fundamentally related by two distinct approaches that arise when learning from data. Many anomaly detection algorithms are concerned with learning/encoding the probability that a sample point will fall within a region $\mathscr{R}$ of input space. As shown in (Duda *et al.*, 2001) is $\mathscr{R}$ is sufficiently small this probability can be approximated by K/NV, with K the number of sample points that fall in $\mathscr{R}$; N the sample size; and V the volume of $\mathscr{R}$. Then, two possible methods of estimating K/NV are: fix V and count the number of points, K, that fall in $\mathscr{R}$; or fix K and calculate how large V, the size of $\mathscr{R}$, must be to contain K points. If we follow the first method we arrive at histogram and Parzen-window density estimation techniques which are related to our T cell approach. If we follow the second method we arrive at k-nearest neighbour techniques which are closely related to the B cell approach.

For both systems, the mean-time-to-failure (MTTF) is an important consideration. This is simply the time between when an anomaly (error) is detected and the failure manifesting itself in the system. Clearly, the MTTF needs to be of a suitable time window to allow for corrective action to be undertaken.

In terms of the requirements of an anomaly detection system outlined in Sect. 4.4.2, the adaptive layer can *correlate* by taking into account various signals over varying time windows (some of which comes from the innate system) and identifying the correlation between inputs and errors; *learning* via the incorporation of new knowledge into a pool of "detectors" that affords the system the ability to identify errors that it has not previously seen; *prediction* in that artificial T and B-cells can identify, with a suitable mean-time-to-failure the occurrence of a fault through the early identification of an error and finally *acclimatisation* as the adaptive layer is endowed with the ability to alter the way in which it detects anomalies depending on environmental conditions. In the adaptive layer, as seen in Fig. 4.35, we employ not only anomaly detection, but also a process of learning and tolerisation which affords the adaptive system with the ability to improve over time. As the robot moves and undertakes tasks in its environment, the robot should learn to define what it considers as its *normal* operating states; the robot learns to classify the *normal* range of input and output values when it is performing a task. We now describe at a high-level how the B-cell and T-cell based AIS will operate.

4.4.4.1 Dynamic Clonal Selection Based Artificial Immune System

An AIS which takes its inspiration from discrimination abilities of the B lymphocyte (B Cell) can provide a realisation of anomaly detection problems *type 1* with scope to address *type 2* problems.

Our dynamic clonal selection anomaly detection system for robots, *dynamicCSR*, is based on concepts on a combination of clonal selection and B-cells as outlined in Sect. 4.4.1.4. We adopt an instance based approach, where we maintain a population of *self-detectors* (analogous to B-cells); with each detector encoding for a feature vector of the system that represents a particular *normal state* of a robotic unit (or *self state*) when performing a task. This differs from the biological B-cells which is used to detect non-self antigen.

The *dynamicCSR* approach allows for variability to be introduced into the population of detectors via an evolutionary process based on clonal selection: where new instances are created that are slight variations of previous self-detectors. With this, the AIS affords the robot the ability to identify new errors that it may not have seen before. We now explore in more detail how this would be achieved, taking into account the AIS framework outlined in (de Castro & Timmis, 2002).

Representation
We discuss two aspects of representation: the *antibody* which is a representation of the robot states and the *antigen* a representation of the input values from the robot sensors.

Antibody: An artificial B-cell (*self-detector*) consists of an array of values that describes the state of the system during a particular task. This array of values assimilates to the antigen receptor of the B-cell, which is used to detect the antigen presented to the cell. The antigen for the self detector is the current state of the robot illustrated in Fig. 4.38.

The receptor array, for example, might include, (i) the type of task being undertaken (Task No.) and the power demand required to perform a task (Power Out), (ii) the status of the components and/or modules monitored by the innate layer or $DangerSig_N$; with N being the component and/or module identification number, and (iii) the actuation output when performing the task, $Actuation_M$; where M is the actuation identification number.

We propose the use of a fuzzy approach to the representation to allow for the capturing of the uncertainties inherent in the data. *Definition of fuzzy set:* If U is a set of ordered pairs of elements denoted generically as u and its respective membership function $\mu_A(u)$, then a fuzzy set, A is defined as Eq. (4.23) (Zimmermann, 2001):

$$A = (u, \mu_A(u))|u \in U \tag{4.23}$$

Example of a membership function for a fuzzy set is the Gaussian membership function, Eq. (4.24), where c and σ is the centre and width of the fuzzy set A.

$$\mu_A(u) = \exp(-\frac{(c-u)^2}{2\sigma^2}) \tag{4.24}$$

An example of a self-detector is given in Fig. 4.37.

Task No.	*Power Demand*	$Actuation_0$	....	$Actuation_M$	$DangerSig_0$	...	$DangerSig_N$	Health Measure

Fig. 4.37 An example of a *self-detector*. N is the number of components monitored by the innate layer and M is the number of actuation modules for the robot.

The actuations outputs can either be fuzzy or binary in value, and would indicate if a module is either on or off. Examples of fuzzified outputs are:

- Motor output: fuzzy set 1 with $\mu(speed) = 1$ at $speed = slow$, set 2 with $\mu(speed) = 1$ at $speed = medium$ and set 3 with $\mu(speed) = 1$ at $speed = fast$.
- Tilt angle: fuzzy set 1 with $\mu(tilt) = 1$ at $tilt = 0.0°$, set 2 with $\mu(tilt) = 1$ at $tilt = 22.5°$, set 3 with $\mu(tilt) = 1$ at $tilt = 45.0°$ and set 4 with $\mu(tilt) = 1$ at $tilt = 90.0°$.

Each self-detector includes a measurement of "health", H_d. The value of H_d is calculated at each sampling time point. The $DangerSig_N$ values from the innate layer ($H_M(t)$) when the robot is performing that particular task contribute to the calculation of H_d; N is the identification number of the monitored module. Note that there are two types of *Health Measure* to the adaptive layer:

- Health measure for the system (robot), $H_M(t)$, which provides for the description of health to the system at a sampling time. This is calculated from the $DangerSig_N$ values from the innate layer.
- Health measure for the detector, $H_d(t)$, that provides for an adaptive measure to the self-detector in the pool. This is calculated based on the difference in affinity of the detector with its *Antigen Vector*, ΔH_d. This measure is defined as the *Activity Measure*, A_M and A_M is the derivative of the affinity, A of the antigen towards the self detector over time, $\Delta H_d = A_M = \frac{dA}{dt}$.

  If ΔH_d is positive: indicating a positive increase in $health$ to the detector, then A_M is also positive. This reinforces the *healthy* detector. If ΔH_d is negative, A_M is also negative, thus assimilates to natural cell death occurring to the biological cell. ΔH_d can therefore provide for an adaptive measure to the detector whereby it can be use to quantify if the detector still falls within the steady state region of the system.

  If a self-detector's $H_d \leq 0$, this self-detector no longer represents the *normal* state of the robot, and has a probability of being removed from the pool. The pool is updated by the algorithm described below. Self-detectors with $H_d \geq \Upsilon$ can be considered as stable states of the system.

Antigen: To ensure that the robot is performing within acceptable operational conditions, system vector of the robot, at sampling time t, will be presented as the *antigen* for self detector. Example of the antigen for the adaptive layer is illustrated in Fig. 4.38.

$Task$ $No.(t)$	$Power$ $Demand(t)$	$Actuation_0$ (t)	....	$Actuation_N$ (t)

Fig. 4.38 Antigen Vector.

Affinity Measure

A self detector provides a representation of the system vector of the robot when the robot is performing a task and is in a particular state. If the *Task No.* and *Power Demand* of the antigen (Fig. 4.38) and an antibody (Fig. 4.37) match, affinity, A between an antigen and an antibody is defined as:

$$A = \sum_{i=0}^{N} \frac{a_i}{N} \tag{4.25}$$

with:

$$a_i = 1 - \left| \frac{D_i}{M_i} \right| \tag{4.26}$$

where: D_i = Antigen Vector's $Actuation_i$ - *Self-Detector*'s $Actuation_i$.
M_i = Maximum difference between the two values. For example: if $Actuation_i$ is $[-x,x]$, $M_i = 2x$.
i is the actuation identification for the robot.

If A is close to 0, $0 < A \leq \phi$, a fault may have occurred in the system.

Algorithm and Processes

We employ a dynamic clonal selection process that maintains a population of *self-detectors*. Algorithm 2 outlines the basic process.

DynamicCSR has the ability to predict the occurrence of an error, based on the previous states of the robot. Antigens are constantly compared to self-detectors. If $(H_d \geq \Upsilon)$, $(H_M \geq \alpha)$ and $(affinity_{high} > \phi)$, we say that no anomaly has occurred in the system, else anomaly is detected. In addition, we might also be able to say that module with $DangerSig_x < \theta$ may well be the source of error. *dynamicCSR* has the ability to predict the occurrence of an error by taking into account the rate of change in the health measures associated with the detectors over time, H_d. This affords great flexibility to the robots, as if such an error is predicted, then a message can be sent to the controller informing them that this particular unit might be removed from the organism, or prevented from entering the organism. To allow for flexibility, *dynamicCSR* has to be able to generate new detectors, and *learn* or adapt the population. For this we employ the idea of a *gene library* (Secker *et al.*, 2003; Cayzer *et al.*, 2005). Simply put, a gene library consists of data that we can use for mutations, or to create new detectors. This allows us to restrict possible combinations of new detectors in such as way as only to allow new self-detectors that map into possible real combinations that are possible to observe in the robotic unit. We may employ a number of libraries that map directly to features in the self-detector.

Algorithm 2. Dynamic Clonal Selection Algorithm for Anomaly Detection in SYMBRION/REPLICATOR robots

Input : A = set of actuation outputs; D_s = set of *DangerSig* from the innate layer;Υ health measure threshold for a single detector; α a health measure threshold for a robot and ϕ a binding threshold between an antigen and antibody
Output: M = set of memory self-detectors describing the behaviours of the robot and each detector has a $health$ measure, H_d and is a normalised value in the range of $[0 \ldots 1]$.
Output: $H_M(t)$ = $health$ measure of the robot and is a normalised value in the range $[0 \ldots 1]$)

begin
  `Perform On-line`
  repeat
    $i = 0$
    forall *self-detectors in M* **do**
      Update the $health$ of a detector $(H_d(t))$ according to its activity measure, A_M;
      Determine the best match self-detector ($affinity_{high}$) that has the highest affinity against its *Input*;
      if $(H_d(t) \leq 0.0)$ **then** Remove detector from population; **else** Determine the *weakest* self-detector, $self_{weak}$;
    end
    Save y number of $healthy$ detectors in pool H;
    if $affinity_{high}$ **then**
      Calculate $H_M(t)$ according to the D_s, its best match self-detector affinity measure and its $H_d(t)$ values;
      Update the best match self-detector's $H_d(t)$ as a function of the $H_M(t)$;
      if *((*$H_M \geq \alpha$*) and (*$affinity_{high} > \phi$*)* **then**
        No anomaly is detected;
        Reward this detector for correct classification of self;
      else
        Anomaly is found;
        Punish detector;
      end
    end
    if *(If no match between Input and detectors)* **then**
      New $self$ is detected;
      Create a new $self$ detector;
      Calculate $H_M(t)$ according to D_s and A_M values;
      Calculate $H_d(t)$ of the new detector as a function of the $H_M(t)$;
      if *((*$H_M \geq \alpha$*) and (*$affinity_{high} > \phi$*)* **then**
        No anomaly is detected;
        Save the new detector to the pool;
      else Anomaly is found;
    end
    Clone and mutate randomly selected self-detector(s) in H, SD_{mutate}
    if *(M has no empty self-detector)* **then** Replace SD_{mutate} with self-detector with $self_{weak}$
    else Save SD_{mutate} to an empty detector in M
  until *forever*
end

Diversity in the pool, therefore, can be achieved by cloning the healthiest self-detector, detector with $H_d \geq \Upsilon$, and performing mutation on the cloned detector. Mutation is performed on the value for one (or more than one) randomly selected $Actuation_y$ variable(s) in the detector and on the value for the *Power Demand* variable, using the gene library as a basis for selecting new data to be inserted in the self-detector. The mutated values are selected from its respective immune gene library, with the values selected are from another healthier detector, detector with $H_d \geq \Upsilon$; irrespective of its *Task No.* value.

All detectors generated must go through a process of *tolerisation* to ensure that the detector is still useful for defining the state of a robot. This is important because the set of *self*-detectors is built online and throughout the life of the robot. This might well be driven by feedback from the robot controller as to the usefulness, or otherwise, of the prediction and/or detection. A new self-detector is generated when $affinity = 0$ and $H_M(t)$ is greater than a threshold value, $H_M(t) \geq \alpha$. To help ensure that an acceptable size of the pool is constantly maintained, *self detector* with $H_d \leq 0.00$ after A_M is deleted from the pool and all its variables values are emptied.

When an anomaly is detected, the *recommendation of response* is produced by selecting the best *"identified"* response from the pool of self detectors. The best *"identified"* response is the actuation values provided in variables $Actuation_y$ that has the closest matching $DangerSig_x$ values but of a better H_d value.

This process is continual, with new detectors being generated, old ones being removed. The effect is a dynamic population of detectors that affords a predictive ability to the robotic units.

4.4.4.2 T Cell Signalling Inspired Anomaly Detection

An AIS which takes inspiration from discrimination abilities of the T lymphocyte (T Cell) can provide a realisation of anomaly detection problems *type 1* and *type 2*.

The T cell (another important agent in the adaptive immune system) must discriminate between "self" and "non-self" molecules on the surface of a antigen-presenting cell (APC). The discrimination abilities are remarkable, a T cell can reliably detect "non-self" despite this signal comprising only $0.01 - 0.1\%$ of the total presented by an APC (Germain & Stefanova, 1999). The discrimination is performed via the T Cell Receptor (TCR). There is evidence that this discrimination ability is partially explained by the specificity of the TCR and partially explained by the complex and *tunable* information processing pathways emanating from the TCR (Germain & Stefanova, 1999).

In (Owens *et al.*, 2010) Owens et al. develop computational models of the TCR signalling pathways which elucidates the T Cell's discrimination abilities. In (Owens *et al.*, 2009) Owens et al. develop an AIS inspired by this TCR signalling to produce an anomaly detection algorithm which addresses *type 1* and *type 2* problems. The modelling work revealed the conceptually important ideas of TCR signalling. These features are depicted in figure 4.39 as a *generalised receptor*. We will use the term receptor to refer to computational structure that is the generalised receptor and we will use the acronym TCR to refer to the biological T Cell Receptor. In most, the

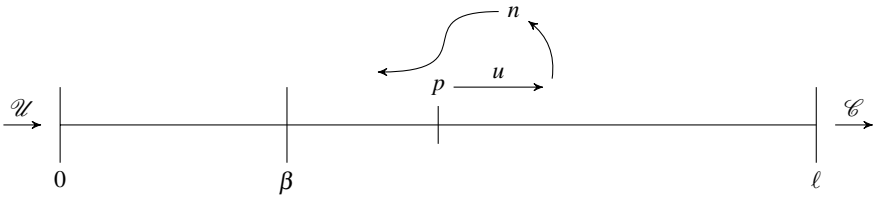


Fig. 4.39 The Generalised Receptor. The receptor receives in input $u \in \mathscr{U}$ and advances the receptor position p. If $p \geq \beta$ the receptor linearly generates negative feedback which arrests and reverses the progression of p. Should $p = \ell$ then the receptor generates an activation signal $c \in \mathscr{C}$.

concepts associated with a receptor are features of the internal (to the T Cell) component of the TCR. A description of the behaviour of a receptor in a computational language suitable for AIS is as follows [2]. A receptor has a position p and a length $\ell > 0$. The initial position is set $p = 0$ and the receptor is said to be activated if p reaches or exceeds ℓ ($p \geq \ell$). The receptor receives an input $u_t \in \mathbb{R}^+$ at time t, the receptor's position will be advanced by a function of u_t. A receptor has a *negative feedback barrier*, β, if $p \geq \beta$ the receptor will linearly generate negative feedback through variable n. The negative feedback will arrest and reverse the progression of p. The negative feedback is generated proportional to the time period that $p \geq \beta$ and not the displacement of p above β ($p - \beta$). The receptor position p and the negative feedback n are also subject to a decay, without input they will both return to zero. A common behaviour, subject to appropriate input, is that the negative feedback will rise to hold the receptor position at equilibrium point $p = \beta$.

In (Owens *et al.*, 2009) it is shown that receptors can be used to exactly reconstruct a conventional anomaly detection technique on static data, namely Kernel Density Estimation (Silverman, 1986) applied to Bayesian Classification (Bishop, 1994). This technique exactly solves *type 1* anomaly detection. Further (Owens *et al.*, 2009) shows that receptors may be used to address *type 2* problems when presented with dynamic data. We provide an overview of this *type 2* approach:

In a system of n sensors each of which provide a real input, we write that inputs $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots \in \mathbf{x} = \mathbb{R}^n$ occur at times $t_1, t_2, t_3, \ldots \in \mathbb{R}^+$ respectively. We define a receptor everywhere in input space, the receptor position at point $\mathbf{x}$ at time t is written as $r_p(\mathbf{x}, t)$ and the negative feedback at $\mathbf{x}$ and t is similarly written $r_n(\mathbf{x}, t)$. We employ two important ideas from biology to determine the update of r_p and r_n in response to an input u_t. The input at position $\mathbf{x}$ will stimulate the receptor at $\mathbf{x}$ and receptors in a neighbourhood around $\mathbf{x}$; the negative feedback generated by a receptor at $\mathbf{x}$ will spread to dampen those receptors in an appropriate neighbourhood around $\mathbf{x}$. Due to the continuous distribution of receptors a kernel function provides a sensible description of the neighbourhood surrounding a receptor. The kernel function should be symmetric, non-negative and integrate to one, the standard Gaussian

[2] For detailed biological descriptions of the behaviour of the TCR see (Owens *et al.*, 2009), (Owens *et al.*, 2010)

kernel is an example. Then $r_p(\mathbf{x},t)$ and $r_p(\mathbf{x},t)$ can be described by the following recurrences:

$$\begin{aligned} r_p(\mathbf{x},t+1) &= r_p(\mathbf{x},t)\phi_b + \Delta(S(\mathbf{x},\mathbf{x_t}) - ar_n(\mathbf{x},t)) \\ r_n(\mathbf{x},t+1) &= r_n(\mathbf{x},t)\phi_d + \Delta g H(r_p(\mathbf{x},t) - \beta) \end{aligned} \qquad H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{4.27}$$

Here $\phi_\alpha \equiv 1 - \Delta\alpha$; $S(\mathbf{x},\mathbf{x_t})$ is an appropriately scaled kernel function; b is the receptor position decay rate; d is the negative feedback decay rate; a is the negative feedback efficacy; g is the negative feedback generation rate; β the negative feedback barrier; Δ the time step granularity; $H(x)$ is the Heaviside step function.

A *type 2* anomaly can be detected if $r_p(\mathbf{x},t) \geq \ell$, see (Owens *et al.*, 2009) for details of why this condition is appropriate. An intuitive explanation is as follows: the rates associated r_p and r_n dictate how quickly the underlying distribution of $\mathbf{x}$ may change. If the distribution changes too rapidly the negative feedback at certain locations may not have enough time to build to contain $r_p(\mathbf{x},t) < \ell$, and so a receptor activation will occur. If the distribution changes at or below the desired rate then the negative feedback will adapt to always ensure that $r_p(\mathbf{x},t) < \ell$. The intuition also explains *type 1* anomaly detection. Regions of input space in which the underlying input distribution is below a threshold receive little or no stimulation. As a consequence the receptor positions will never break the negative feedback barrier and these regions will have no negative feedback. The parameters may then be set such that an input to a receptor with zero negative feedback will result in the receptors position exceeding ℓ. Thus an input in a below threshold region will cause a receptor activation which is the condition for *type 1* anomaly detection.

4.4.4.3 Sharing of Immunological Information

One of the main drives behind the SYMBRION/REPLICATOR is to allow for robotic units to share resources, thus allowing for the organism to function effectively. With this in mind, we propose to allow for the sharing of "immunological" information between units via an artificial lymphatic system. In the previous sections we have discussed a combined innate and adaptive artificial immune system capable of affording fault tolerance. To promote greater fault tolerance to the collective robotic system such information should be shared. The artificial lymph node contained within a robot detects an anomaly to its internal state, this is achieved either at the innate, or adaptive level. It then provides an appropriate response to the high level controller. If a beneficial action has been undertaken, the "immunological" information that was responsible for that should be shared with neighbouring (or connected) robots. If this is to be undertaken, then care needs to be taken to ensure that new "immunological" information does not overwrite, older, but still potentially useful information.

This can be achieved in a similar manner to how network wide learning was achieved in work by (de Lemos *et al.*, 2007), where a collaborative system was proposed, in the context of a network of automated teller machines (ATM's) where

Table 4.5 Properties of granuloma formation

Properties of swarm robotics	Properties of Granuloma Formation
Large number of robots	Large number of cells
Few homogenous groups of robots	Few homogenous cells
Relatively incapable or inefficient robots	Each cell needs each other to perform the desired task
Robots with local sensing and communication capabilities	Chemokines, cytokines

a second pool of 'immunological" information was maintained, in that case, this was done centrally, something that can not be implemented in a collective robotic system. In this second pool, known as the pool of *network detectors*, if similar detectors were also present or observed over a period of time (these are detectors that were sent in from local error detection systems on an individual ATM) then this was taken to be an indicative error detector and shared with other ATM's in a logical or physical neighbourhood. A similar process could be employed on the robotic units. A secondary immunological memory can be maintained, that stores either danger signal indicators from the innate AIS; detectors from the *dynamicCSR* algorithm, or receptor parameters from the T Cell AIS. A period of *tolerisation* is required to ensure that only relevant and useful information is preserved, and a special guard placed on the transferring of information from this secondary pool to the primary pool needs to be in place.

With this system in place, the robotic collective will be able to learn about new errors both at a local and also a collective level, affording a greater degree of fault tolerance across the collective.

4.4.4.4 Aggregation of Robots

Within our proposed AIS approach, not only will we be able to detect errors and make recommendations as outlined above, which operate at the individual robot level, we can also afford the swarm a collective behaviour of aggregation in order to isolate a faulty robot, and in some cases allow for the repair of the robot, both when operating in swarm mode or organism mode: this affords a *self-healing* property on the system. For this, we take our inspiration from an immunological process known as *granuloma formation*. Granuloma formation is a complex process involving a variety of mechanisms acting in concert to bring an inflammatory lesion that is able to contain and destroy intracellular pathogens. While it is crucial to host defence, inappropriate granulomatous inflammation can also consider as damage to host defence. The main actors in granuloma formation are: macrophages, T-cells and cytokines. Granuloma formation is comprised of four main steps (Sneller, 2002):

- The triggering of T cells by antigen presenting cells, represented by certain types of macrophages and dendritic cells.
- The release of cytokines and chemokines by such macrophages, which are essentially activated lymphocytes and dendritic cells. Cytokines and chemokines attract and retain cell populations in the area of activity, inducing their survival and proliferation at the site of ongoing inflammation.
- The cytokine release results in the stable and dynamic accumulation of immunocompetent cells (cells that can act in an immune response) and the formation of an organised structure of the granuloma.
- The last phase of granuloma formation generally ends in fibrosis (large amounts of extra tissue which could be harmful). Granuloma formation begins when an infectious diseases enters a host.

Macrophages will *"eat"* or engulf bacteria to prevent the bacteria spreading, however, the bacteria will infect macrophages and duplicate as much as possible. Thus, despite the macrophages ability to stop the infections, bacteria will use macrophages as a *"taxi"* to spread disease within the host leading to the cell lysis (death) or breaking down the structure of the cell. Infected macrophages will emit signals indicating that they are infected. This signal will attract other macrophages to the site of infection, and form a barrier around the infected macrophage by isolating the infected cells. This ultimately leads to the formation of a either a chronic granuloma (one that persists over time and is not harmful to the host); a healing scenario where the infection is removed from the host; or in the worst case the granuloma continues to expand and will harm the host potentially leading to death.

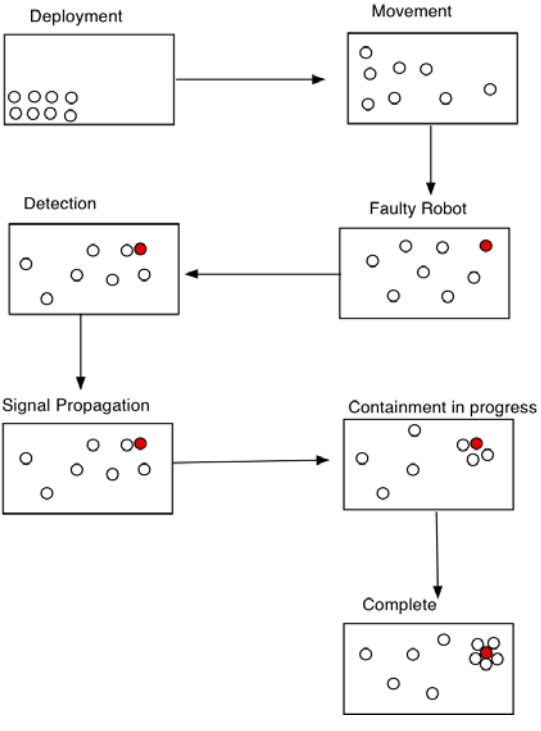


Fig. 4.40 Simple Scenario: Formation of robots

In granuloma formation, macrophages form a *wall* around the chronically infected macrophages as well as the bacteria leading to the formation of granuloma with the objective of separating the infected and uninfected cells as well as

separating the bacteria from the healthy cells. By separating the infected macrophages, the robustness of the system will be maintained and the failure of one or more cells should not affect other cells and the overall operation of the system. We propose to take this idea into swarm and organism robotics. Key actors in the formation of granulomas are macrophages, cytokines and chemokines which act as the signalling mechanisms and T-cells.

Applying the granuloma formation concepts to collective robotics allows us to perform two related tasks. First, we may isolate faulty robots. Given that above we have described how we detect faulty robots, during swarm mode this may effect the performance of other robots and act as an "anchor" to other robots when they are moving. This occurs due to the local signalling mechanism employed by the robots to allow them to navigate around the environment. Fig. 4.40 shows different phases of aggregation used to isolate the faulty robot based on concepts from granuloma formation.

Consider the case when a permanent fault is located in a robot, and the robot ceases to move. We can assume that certain visual signals can be sent by the robot which other functional robots nearby can recognise. These functional robots are then attracted towards the faulty robot, akin to how T-cells are attracted by cytokines emitted by an infected macrophage. A limited number of these robots then isolate the fault robot, akin to T-cells surrounding an infected macrophage, but still move around the fault robot so that other functional robots in the swarm are no longer drawn to the "anchor" point that could be the faulty robot. This approach would be ideally used when certain repairs could be initiated by the robots themselves. Consider the case then a transient fault occurs in robot that results in a large power drain in the robot. What is required now, is for other robots to share power with each other to re-charge the "faulty" robot. Employing the approach above allows us to surround the faulty robot with functional robots that are able to share power. In SYMBRION/REPLICATOR robots, such power sharing is possible. So, once the faulty robot is surrounded, then power sharing can being. Once a power share operation has been completed, then robots in artificial granuloma will then carry on with the tasks they were doing before the fault was identified. This extra layer of fault tolerance at the collective level, allows for a fully integrated fault tolerant system in the robots, something that is unique to any collective robotic system.

4.4.5 Summary

In this chapter we have proposed an integrated Artificial Immune System for fault tolerance in collective robotic systems. Fault tolerance in swarm and collective robotics has not received the attention that is should in the past, and we see that for such systems to be truly successful they need to be able to cope with changing environments and the presence of faults. We proposed error detection in these systems in the context of anomaly detection, and defined three types of anomaly: types 0, 1 and 2 all of which have different "signatures". Type 0 are associated with deviations from known constraints on sensors; type 1 are associated comparisons between current sensor readings and a recent history of sensor readings; type 2 are associated

with macroscopic changes in the history of sensor readings. Our integrated AIS approach allows us to identify all three types of anomaly. Our innate AIS identifies type 0 anomalies by looking for deviations from known operational boundaries of sensor values. This artificial innate immune system also provides vital information for one aspect of our adaptive AIS. At the adaptive level, we have a two pronged approach, one instance based anomaly detection system based on ideas taken from clonal selection theory in immunology, and the second based on the tuning ability of T-cells, which is translated to a statistical anomaly detection system. Both of these systems are able to identify types 1 and 2 anomalies. Through the dual approach, we engineer in redundancy into the fault tolerant AIS, so if one arm of the adaptive AIS does not identify a potential error, the second will. Both of the adaptive systems are capable of learning about new and different anomalies in an automatic manner: thus affording a great deal of flexibility to the robotic units. In addition to the identification of errors on a robotic unit, we have discussed how not only those units can share learnt information, thus allowing the collective of robots to improve individual immune systems, but also how ideas from the formation of granulomas in the natural immune system can act as a metaphor for the aggregation of robots to isolate and repair faulty robots, thus reducing the impact of failing robots on overall collective behaviour. Once realised, this AIS will be the first fully integrated bio-inspired fault tolerant system for collective robotics, which will allow for greater flexibility and autonomy of the collective robotic system than was previously possible.

4.5 Structural Self-organized Control

Serge Kernbach, Olga Kernbach

Running evolutionary algorithms for evolving morphology and functionality has an advantage of finding very specific solutions in a large search space. As demonstrated in this book, this approach is useful for evolving topologies, kinematics and control without a human assistance in a priori unknown situations. However, environments normally contain many known and unknown components. For such situations, which have more known than unknown elements, the task is rather to select and to adapt one of pre-evolved (or pre-developed) solutions instead of evolving the required topology and functionality anew.

Combination of off-line pre-development and on-line selection/adaptaton of structural solution has several advantages. Firstly, different topologies require kinematic, controlling, homeostasis, energetic and many other mechanisms. Evolving a new organism often means not only a new topology, but also new control and regulatory elements. This requires multiple tests and resources such as time or energy. Performing this development on-line can seriously damage the system, for example, evolving the collision avoidance by using physical contacts with other objects can mechanically destroy the system. Embodied evolving is normally done in a test environment, which possesses enough required resources, is less constrained and prevents the system from damages. Secondly, due to technological constraints

(e.g. a maximal weight of organism or reasonable kinematics) and different symmetries the realistic search space for middle-size organisms (up to 20-25 modules) is not so large. As shown later in this section, only the centipede-like, planar fungi-like and wheeled topological generators (see Tables 4.6 and 4.7) cover a large number of scalable topologies, representing an essential part of the structural search space.

More generally, the set Φ of all possible topologies can be split into three different subsets: the subset of pattern-based topologies Φ^P, the subset of forbidden/not-efficient topologies Φ^υ, and finally the subset of specific topologies Φ^S. The largest group represents the pattern-based subset Φ^P. It consists of variations of *a priori known* efficient patterns ϕ_i, such as caterpillar-like, dragon-like, 4- and 6-legs-like, or wheeled topologies. When we allow a perturbation of the pattern by at least 1 module (i.e. one module in the dragon-like shape can be connected randomly – there are 20 possible dragon-like shapes) perturbations of these 5 patterns cover over 100 possible structures of an organism. Forbidden and not-efficient combinations Φ^υ appear when some constraints υ_i are not satisfied. For example, when the modules are connected so closely to each other that they can mechanically be destroyed, this configuration is forbidden. The set of specific combinations Φ^S represents such topologies, which are *unusual-but-efficient*, typically they are result of evolving in specific environments. Due to the generators *structure* $\rightarrow$ *function* $\rightarrow$ *behavior*, the impact of structures on functionality and behavior is in the loop of improvements – this represents an advantage of Φ^S.

The idea of this section consists in the following observation. The set of pre-generated patterns Φ^P can be maximized so that to cover the most functionally useful behaviors in different predictable environmental situations. For example, this can be performed by off-line evolutionary simulation, by following a bio-inspiration from insect or animals and in general is done *in advance*. These patterns can structurally be perturbed by a few modules *on-line*. This perturbation creates some deviation in the expected functionality and behavior, which can be handled by different on-line adaptation mechanisms. Then, we assume that a subset of specific topologies Φ^S (with such an on-line adaptation) is *almost-functionally-equivalent* to the subset of *perturbed pre-generated patterns* Φ^P.

In this way, the problem of evolving a specific solution is replaced by the problem of optimizing a deviation from one of the pre-generated patterns, for which all controlling mechanisms exist. Since linear optimization is very fast, for example, the linear sum assignment problem is of $O(n^3)$ complexity (Burkard *et al.*, 2009), this approach can be run on-board and on-line. Moreover, optimization can be considered as a mean of synchronization between different modules (i.e. two independent optimizers receive the same results when they use the same initial data). This allows us to use self-organizing mechanisms for a structural regulation.

Further we focus on two issues: firstly, effective representation of topologies and their perturbations, on-line generation by a structural generator and the related points of scalability (Sect.4.5.1 – Sect. 4.5.3) and, secondly, on-line distributed self-assembling (Sect. 4.5.4 – Sect. 4.5.5). Both issues complement each other. Finally, in Sect. 4.5.6 we discuss collective memory and draw some conclusions.

4.5.1 Representation of Structures

Since assembling and disassembling is performed on a 2D plane (due to mechanical stress in docking elements, see more in Sect. 2.1), moreover since each docking element can be connected only to one another docking element, most topologies of artificial organisms generally belong to 2D grid-based reconfigurable systems. The matrix-based (and correspondingly a graph-based) representation of such topologies is common in reconfigurable robotics, see e.g. (Chiang & Chirikjian, 2001), (Salemi & Shen, 2004) or (Lau *et al.*, 2008).

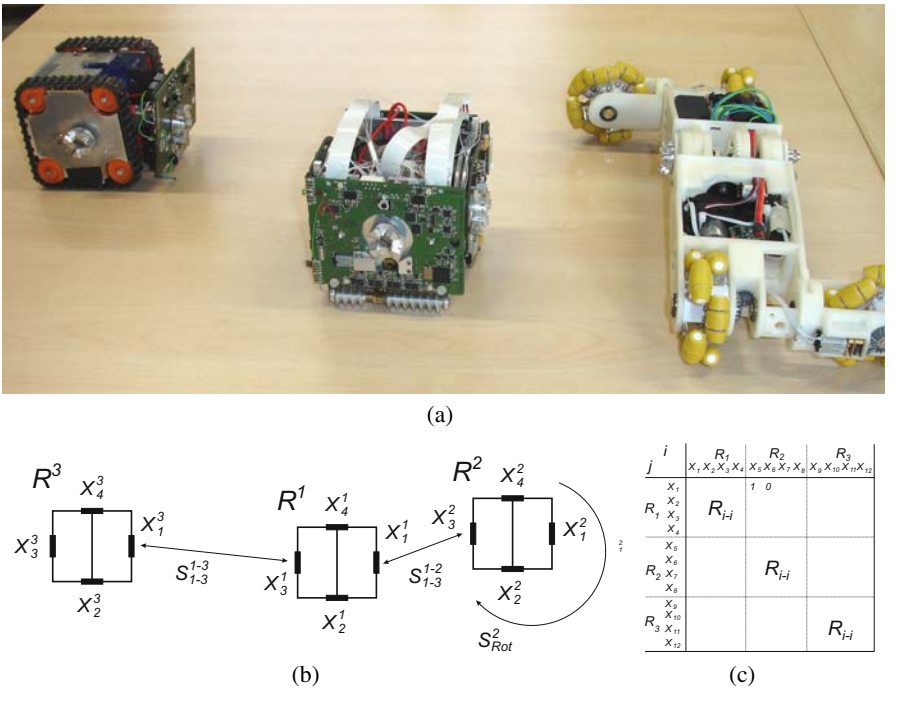


Fig. 4.41 **(a)** Example of three robot modules going to aggregate into the organism, **(b)** Schematic representation, **(c)** Initial topological matrix $\underline{\underline{C}}$.

Let us first consider the small case of three different robot modules, see Fig. 4.41, which are going to connect to each other and to create a small organism. Each module has a different number of docking elements and has different capabilities. We denote *placeholders* for robot modules as R_r^t, where the upper index t denotes the type of a robot, the lower index r - a sequential number of a robot. The index r is mainly used in topological calculations, where the index t indicates functionality of

modules. Each docking element will be denoted as x_i^r, where the upper index indicates a robot r and the lower index i indicates a docking element. All i are numbered clockwise as "1" - front, "2" - right, "3" - back and "4" - left connector. In general, this notion can be chosen in another way, see e.g. in Sect. 2.4, it does not change the following idea of generators.

When we place all x_i sequentially, a topology of the organism can be represented by the adjacency matrix $\underline{\underline{C}}$, as shown in Fig. 4.41(c). This is a low-level representation of the topology and this matrix has several properties. Firstly, elements of the matrix $x_{i,j}$ are integers, which indicate connections and rotations between elements. For example, "1" between x_1 and x_5 means that R_1 and R_2 are connected via x_1^1 and x_1^2 without rotation; the "0" between x_1 and x_6 means that R_1 and R_2 are not connected via x_1^1 and x_2^2. The "2", "3" or "4" indicates that corresponding element is rotated clockwise by 90°, 180° and 270°. The "1" on the main diagonal means that x_i is connected with itself, i.e. not connected with any other docking element. Thus, the adjacency matrix $\underline{\underline{C}}$ can represent both the connection and the rotation between elements.

The adjacency matrix can be essentially simplified by considering symmetries of modules (i.e. equal functionality at 90° and 270°). However, all modules have different symmetries and to generalize their treatment we assume that the same module, but rotated in different ways, represents different robots. This allows a separation between functionality and connections of modules and also makes more evident a transition from topologies to kinematics. Thus, $\underline{\underline{C}}$ can have only "0" or "1" and each x_i can be connected only one time, i.e.

$$x_{i,j} = \begin{cases} 1 & \textit{if } x_i \textit{ is docked to } x_j, \\ 0 & \textit{otherwise.} \end{cases} \qquad \sum_i^n x_{i,j} = 1, \quad \sum_j^n x_{i,j} = 1. \tag{4.28}$$

Now we take look at the square elements on the main diagonal of $\underline{\underline{C}}$. There are 16 combinations, which will be denoted as A (connectivity four), B (connectivity three), C (connectivity two) and D (connectivity one). Zero sub-matrix is denoted as 0.

$$\begin{matrix}1000\\0100\\0010\\0001\end{matrix}=A,\quad \begin{matrix}0000\\0100\\0010\\0001\end{matrix}=B_1,\quad \begin{matrix}1000\\0000\\0010\\0001\end{matrix}=B_2,\quad \begin{matrix}1000\\0100\\0000\\0001\end{matrix}=B_3,\quad \begin{matrix}1000\\0100\\0010\\0000\end{matrix}=B_4,$$

$$\begin{matrix}1000\\0100\\0000\\0000\end{matrix}=C_1,\quad \begin{matrix}1000\\0000\\0010\\0000\end{matrix}=C_2,\quad \begin{matrix}1000\\0000\\0000\\0001\end{matrix}=C_3,\quad \begin{matrix}0000\\0100\\0010\\0000\end{matrix}=C_4,\quad \begin{matrix}0000\\0100\\0000\\0001\end{matrix}=C_5,$$

$$\begin{matrix}0000\\0000\\0010\\0001\end{matrix}=C_6,\quad \begin{matrix}1000\\0000\\0000\\0000\end{matrix}=D_1,\quad \begin{matrix}0000\\0100\\0000\\0000\end{matrix}=D_2,\quad \begin{matrix}0000\\0000\\0010\\0000\end{matrix}=D_3,\quad \begin{matrix}0000\\0000\\0000\\0001\end{matrix}=D_4 \tag{4.29}$$

Square 4×4 sub-matrices, which are not on the main diagonal, have an additional property $\sum_{i,j}^{4} x_{i,j} = 1$, i.e. they contain always only one "1" element in the whole sub-matrix. There are also only 16 such elements, which will be denoted as Z_i.

$$\begin{matrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_1, \quad \begin{matrix}0&1&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_2, \quad \begin{matrix}0&0&1&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_3, \quad \begin{matrix}0&0&0&1\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_4, \quad \begin{matrix}0&0&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_5, \quad \begin{matrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_6,$$

$$\begin{matrix}0&0&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&0\end{matrix} = Z_7, \quad \begin{matrix}0&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{matrix} = Z_8, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\1&0&0&0\\0&0&0&0\end{matrix} = Z_9, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&0\end{matrix} = Z_{10}, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\end{matrix} = Z_{11}, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{matrix} = Z_{12},$$

$$\begin{matrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\1&0&0&0\end{matrix} = Z_{13}, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&1&0&0\end{matrix} = Z_{14}, \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{matrix} = Z_{15} \quad \begin{matrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{matrix} = Z_{16}$$

All Z_i are symmetrical, i.e.:

$$Z_2^T = Z_5, \;\; Z_3^T = Z_9, \;\; Z_4^T = Z_{13}, \;\; Z_7^T = Z_{10}, \;\; Z_8^T = Z_{14}, \;\; Z_{12}^T = Z_{15}, \tag{4.30}$$

where Z^T means a transpose of Z. All Z_i with "1" on the main diagonal are the same, i.e. $Z_i^T = Z_i$. Relations (4.30) are useful in finding symmetrical representation of topologies. This basic symbolic representation is similar to (Castano & Will, 2001). In Fig. 4.42 we show a few examples of simple structures and their symbolic representations. We can intuitively say, the more regular is the structure of the organism, the more compact is the matrix representation of this topology. The macro-wheel, shown in Fig. 4.42(a), has the most compact representation. When we denote as M the $n \times n$ square matrix, the macro-wheel from n modules has the circulant form (Davis, 1979), defined by

$$\underline{\underline{C}}^n_{macro-wheel} = M(circ\{C_5, Z_9, 0, 0, 0, ..., Z_9^T\}). \tag{4.31}$$

The caterpillar (snake) from Fig. 4.42(b) is in fact a band matrix, whose first and last elements are perturbed. We can represent this topology as a difference between a regular band matrix and corresponding deviation

$$\underline{\underline{C}}^n_{sn} = M(band\{Z_9^T, C_5, Z_9, 0, 0, 0, ..., 0\}) - M(\{\underline{\underline{x}}_{1,1} = D_1, \underline{\underline{x}}_{n,n} = D_3\}). \tag{4.32}$$

The (4.32) can be generalized as

$$\underline{\underline{C}} = Regular\ Matrix \pm Deviation\ Matrix. \tag{4.33}$$

The (4.33) represents a general way of dealing with basic topologies – each basic topology can be represented as some regular pattern and some perturbation. This also means that a topology can have several representations. To be more ordered in the notation, we propose to use the following numeration rules:

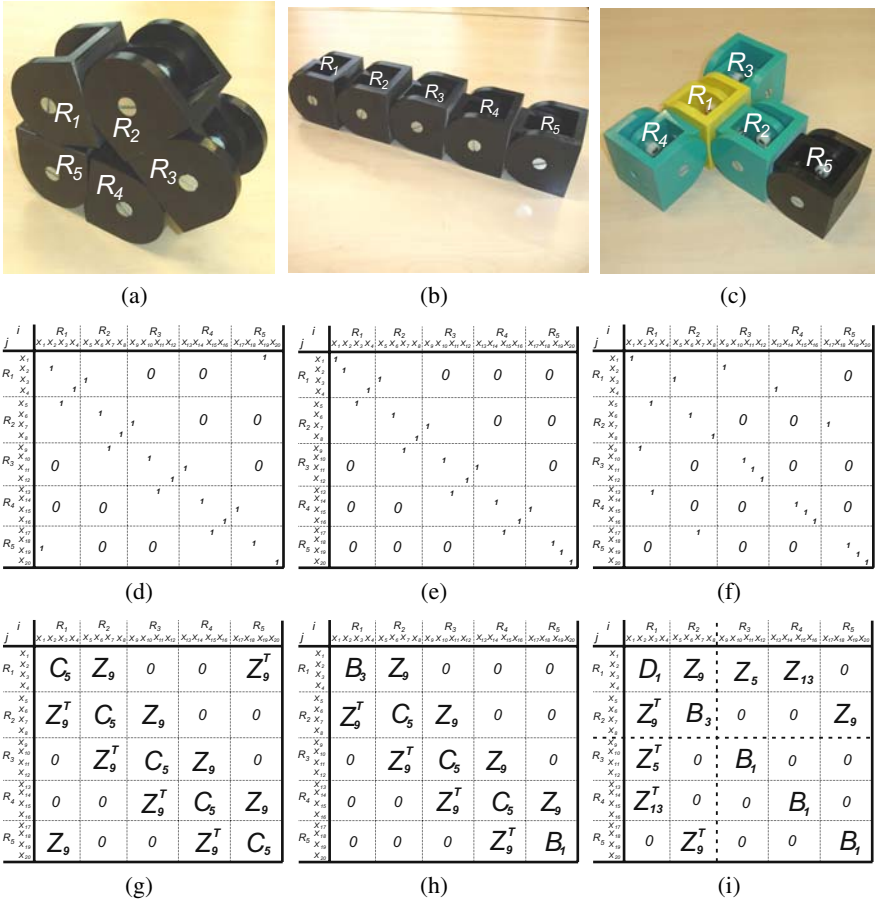
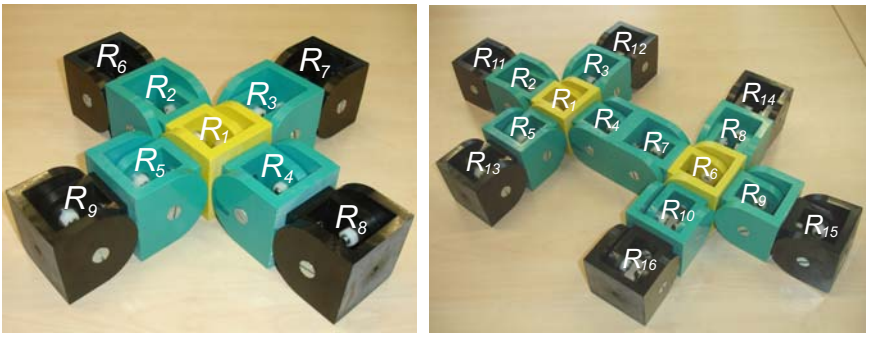


Fig. 4.42 Different topologies of the organism and their representations, see also Table 2.9. **(a), (d), (g)** Macro-wheel, where all modules have two connections; **(b), (e), (h)** Caterpillar (snake), where the R_1 and R_5 are marginal (connected only one time); **(c), (f),(i)** Dragon, where R_3 and R_4 are connected to R_1.

1. Firstly such elements are numbered, which have a maximal degree of connectivity;
2. All other elements are numbered in order of decreasing their connectivity and increasing a distance to the first elements;
3. Lastly all marginal (closing) elements are numbered.

For example, "the dragon" from the Fig. 4.42(c) is numbered by following this scheme. As we can see, the symbolic representation in Fig. 4.42(i) has a well-structured form: the lowest square matrix of marginal $R_3 - R_5$ elements is a diagonal one. We can extend this idea for larger topologies with 9 and 16 modules, shown in Fig. 4.43. Here we consider a symmetric cross and 2x-Centipede ("dog"), made



(a) (b)

j \ i	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}
R_1	0	Z_1	Z_5	Z_9	Z_{13}											
R_2	Z_1^T	C_5				Z_9										
R_3	Z_5^T		C_5				Z_9									
R_4	Z_9^T			C_5				Z_9								
R_5	Z_{13}^T				C_5				Z_9							
R_6		Z_9^T				B_1										
R_7			Z_9^T				B_1									
R_8				Z_9^T				B_1								
R_9					Z_9^T				B_1							
R_{10}																
R_{11}																
R_{12}																
R_{13}																
R_{14}																
R_{15}																
R_{16}																

Virtual robots

(c)

j \ i	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}
R_1	0	Z_1	Z_5	Z_9	Z_{13}											
R_2	Z_1^T	C_5									Z_9					
R_3	Z_5^T		C_5									Z_9				
R_4	Z_9^T			C_5			Z_{11}						Z_9			
R_5	Z_{13}^T				C_5											
R_6						0	Z_1	Z_5	Z_9	Z_{13}						
R_7			Z_{11}^T			Z_1^T	C_5									
R_8						Z_5^T		C_5						Z_9		
R_9						Z_9^T			C_5						Z_9	
R_{10}						Z_{13}^T				C_5						Z_9
R_{11}		Z_9^T									B_1					
R_{12}			Z_9^T									B_1				
R_{13}				Z_9^T									B_1			
R_{14}							Z_9^T							B_1		
R_{15}								Z_9^T							B_1	
R_{16}									Z_9^T							B_1

(d)

Fig. 4.43 Examples of 9x and 16x topologies. **(a)** Extended cross; **(b)** 2x-Centipede ("dog"), obtained as a combination of two extended crosses.

as a combination of two crosses. Their corresponding symbolic representations are shown in Figs. 4.43(c),(d). Basically, it repeats the pattern from Fig. 4.42(i) – strong core elements and weak marginal elements on the main diagonal, as well as connections between them on the upper right sector. The lower left sector is just a symmetry of the topology matrix. Exploring the symmetry of matrix models, see more in (Flener *et al.*, 2002) or (Kiziltan & Milano, 2002), we can draw conclusions about topology of organisms even when different notation systems has been used (e.g. before and after merging two structures).

The notations shown in Figs. 4.42, 4.43 involve placeholders R_i instead of real robot ID's. In the morphogenetic process, we have to map real robot ID's into the placeholder R_i ($ID_j \rightarrow R_i$) and to solve the appearing constrained assignment problem. This approach is treated in Sect. 4.5.4, where we demonstrate several self-organizing properties of that approach. In the next section we consider more compact representation of the topology with topological generators.

4.5.2 Compact Representation: The Topology Generator

As demonstrated by (4.31) or (4.32), the topology can be represented as a compact generator. This representation has many advantages, such as a low memory consumption, low communication effort, possibilities of topological analysis. The idea to introduce a generator is not new, see e.g. Sect. 2.4 with an example of L-systems or the work of (Brener *et al.*, 2008) for an application of the group-based approach. For a generator, we involve not only a well-defined group-based formalism, but also several pragmatic considerations.

The first pragmatic consideration is related to a structural stability of reconfigurable systems. To provide a mechanical stability, several elements should be strongly connected, in terms of multiple connections between them. We denote such an aggregation as a strong core. The strong core is clearly visible in Fig. 4.43(d) - elements $R_1 - R_5$ and $R_6 - R_{10}$ builds two strong cores. Making the system larger, but still structurally stable, requires insertion of more core elements – this reflects the structural scalability, see Sect. 4.5.3. Strong cores should be connected with each other and with weakly connected marginal elements. This idea is represented in Fig. 4.44.

Now we denote the core matrix as $\underline{\underline{C}}_c$ and the marginal matrix as $\underline{\underline{C}}_m$, as well as $core_i$-$core_j$ coupling as $\underline{\underline{C}}_{c,c}$ and $core_i$-$margin_j$ couplings as $\underline{\underline{C}}_{c,m}$. The topology of an organism can be represented as:

$$\underline{\underline{C}} = M(\underline{\underline{C}}_c + \underline{\underline{C}}_m + \underline{\underline{C}}_{c,c} + \underline{\underline{C}}_{c,m}), \tag{4.34}$$

where M acts as an operator, which creates $\underline{\underline{C}}$, see Fig. 4.44.

Each of the matrices $\underline{\underline{C}}_c$, $\underline{\underline{C}}_m$, $\underline{\underline{C}}_{c,c}$, and $\underline{\underline{C}}_{c,m}$ is a low-dimensional matrix, which can be generated in the way of (4.33) i.e. analytically. The whole topology matrix can be then reconstructed by using the pattern from Fig. 4.44. The expression (4.34) is an example of a topological operator, defined over a set of templates $\underline{\underline{C}}_c$ and $\underline{\underline{C}}_m$.

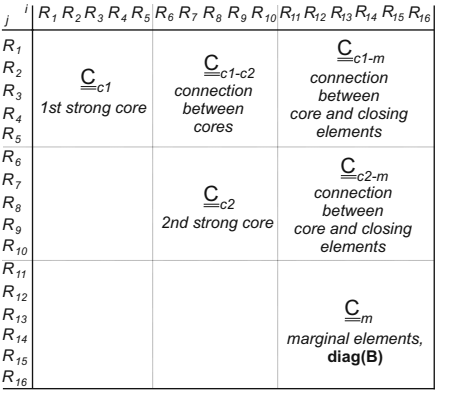


Fig. 4.44 Generalized representation of the topology matrix from Fig. 4.43(c).

The idea of templates and operators can be generalized in the following way. Considering the topology of elementary modules, we can define four basic core elements: 1x-core as one separate module R_i, 4x-core is a T-like shape from Fig. 4.42(c) (when to remove the element R_5), 5x-core is a cross of 5 elements from Fig. 4.43(a) (when to remove $R_6 - R_9$), defined both as:

$$\underline{\underline{C}}_{1xc} = A,\ \underline{\underline{C}}_{4xc} = \begin{matrix} D_1 & Z_5 & Z_9 & Z_{13} \\ Z_5^T & B_1 & 0 & 0 \\ Z_9^T & 0 & B_1 & 0 \\ Z_{13}^T & 0 & 0 & B_1 \end{matrix},\ \underline{\underline{C}}_{5xc} = \begin{matrix} 0 & Z_1 & Z_5 & Z_9 & Z_{13} \\ Z_1^T & C_5 & 0 & 0 & 0 \\ Z_5^T & 0 & C_5 & 0 & 0 \\ Z_9^T & 0 & 0 & C_5 & 0 \\ Z_{13}^T 0 & 0 & 0 & 0 & C_5 \end{matrix} \tag{4.35}$$

and, finally, the caterpillar topology, defined by (4.32). These four cores represent four basic templates. Now, in order to generate different topologies, we need to define several operators over these templates. Generally, the number of possible topological operators can be large and depends on the heterogeneity of modules and practical requirements. Further we define only three operators and show that even this small number of operators can compactly generate a large set of topologies.

Connection (join) operator. This operator connects two different topologies into one structure. We will denote it as $Con(Top_1, Top_2, x^1_{i,j}, x^2_{k,l})$, which connect topologies 1 and 2 between elements $x^1_{i,j}$, $x^2_{k,l}$. Application of this operator can be considered in Fig. 4.43(c) and 4.43(d). It basically changes connectivity of the $x^1_{i,j}$, $x^2_{k,l}$ on the main diagonal and introduce corresponding elements into the connection parts of $\underline{\underline{C}}_{c,c}$. $Con^n()$ means an iterative n-time application of $Con()$. After connecting two topologies, all modules needs to be renumbered, this is normally done by middleware system of a robot organism.

Margining operator. This operator is a symmetric one and creates a margin of robots around an existing structure. We denote this as $Mar(Top_1, x_i, x_j)$.
For example, $Mar(\underline{\underline{C}}_{5xc}, x_3, x_1)$ connects a new robot by using the docking x^i_1 to each free docking element x^i_3 of the $\underline{\underline{C}}_{5xc}$. As a result we receive the shape shown in Fig. 4.43(a). Like in the previous case, $Mar^n()$ means an iterative n-time application of $Mar()$.

DoF swap. Connecting robots in the caterpillar-like way, shown in Fig. 4.42(b), creates actuation only in the vertical plane. To extend this actuation into a horizontal plane, one robot should have a horizontal degree of freedom, i.e. it has to swap its own DoF. The operator $DoF(x_{i,j})$ swaps the DoF of a robot in the position $x_{i,j}$ (e.g. it changes a connection from horizontal to vertical one or vice versa).

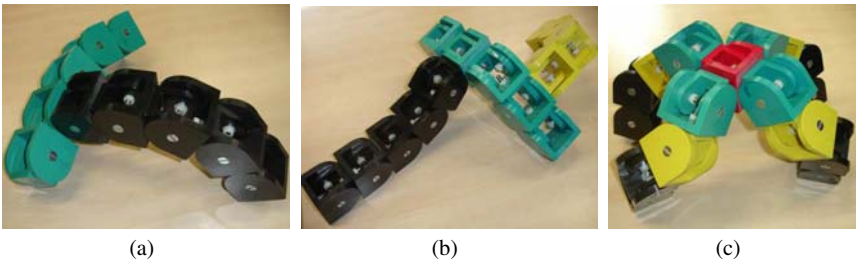
Table 4.6 and Fig. 4.45 demonstrate several generators and examples of generated structures.

4.5.3 Scalability of Structures and Appearing Constraints

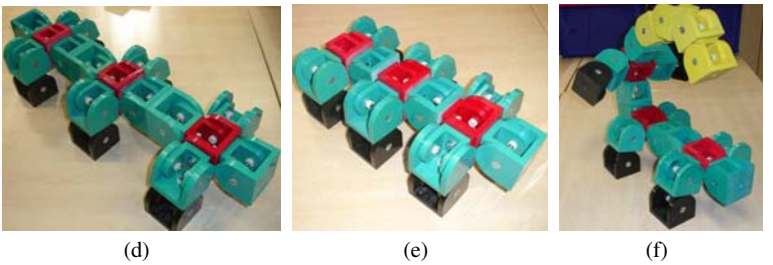
The generator allows considering structures not only as a fixed set of elements, but as a scalable organization. Examples are n-centipedes, shown in Fig. 4.43(b) and Fig. 4.45(d) for 16 and 21 robots, generated by $Mar(Con^n(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}_{5xc}, x^4_3, x^2_3), x_3, x_1)$, and the n-Dragon (5x Core), generated by $Con(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}^n_{sn}, x^3_3, x^1_1)$. Different scalable structural elements, such as $\underline{\underline{C}}^n_{sn}$, or scalable operators, such as $Con^n()$, provide a way of dealing with scalability of structures. Table 4.7 shows several types of structures and their scalability classes.

Table 4.6 Examples of topological generators.

Topology	N	Generator	Example
1x-Core	1	$\underline{\underline{C}}_{1xc}$	
4x-Core	4	$\underline{\underline{C}}_{4xc}$	
5x-Core	5	$\underline{\underline{C}}_{5xc}$	
Caterpillar	n	$\underline{\underline{C}}^{n}_{sn}$	Fig. 4.42(b), n=5
2x Cross-caterpillar	10	$Con(\underline{\underline{C}}^{5}_{sn}, \underline{\underline{C}}^{5}_{sn}, x_8, x_1)$	Fig. 4.45(a)
3x Cross-caterpillar	14	$Con(\underline{\underline{C}}^{4}_{sn}, Con(\underline{\underline{C}}^{5}_{sn}, \underline{\underline{C}}^{5}_{sn}, x_8, x_1), x_{14}, x_1)$	Fig. 4.45(b)
2x QuadroPod	9	$Mar(\underline{\underline{C}}_{5xc}, x_3, x_1)$	Fig. 4.43(a)
3x QuadroPod	13	$Mar^2(\underline{\underline{C}}_{5xc}, x_3, x_1)$	
4x QuadroPod	17	$Mar^3(\underline{\underline{C}}_{5xc}, x_3, x_1)$	
4x QuadroPod sw. DoF	17	$Mar^2(Mar(\underline{\underline{C}}_{5xc}, x_3, x_2), x_4, x_1)$	Fig. 4.45(c)
Macro-wheel	n	$M(circ\{C_5, Z_9, 0, 0, 0, ..., Z_9^T\})$	Fig. 4.42(a)
n-Dragon with 4x Core	n+3	$Con(\underline{\underline{C}}_{4xc}, \underline{\underline{C}}^{n}_{sn}, x_3^3, x_1^1)$	Fig. 4.42(c), n=1
n-Dragon with 5x Core	n+4	$Con(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}^{n}_{sn}, x_3^3, x_1^1)$	
2x-Centipede ("dog")	16	$Mar(Con(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}_{5xc}, x_3^4, x_3^2), x_3, x_1)$	Fig. 4.43(b)
3x-Centipede ($\underline{\underline{C}}_{5xc}$)	21	$Mar(Con^2(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}_{5xc}, x_3^4, x_3^2), x_3, x_1)$	Fig. 4.45(d)
3x-Centipede ($\underline{\underline{C}}_{4xc}$)	18	$Mar(Con^2(\underline{\underline{C}}_{4xc}, \underline{\underline{C}}_{4xc}, x_3^4, x_3^2), x_3, x_1)$	Fig. 4.45(e)
Scorpion ($\underline{\underline{C}}_{5xc}$)	25	$Con(\underline{\underline{C}}^{4}_{sn}, Mar(Con^2(\underline{\underline{C}}_{5xc}, \underline{\underline{C}}_{5xc}, x_3^4, x_3^2), x_3, x_1), x_{54}, x_1)$	Fig. 4.45(f)



(a) (b) (c)

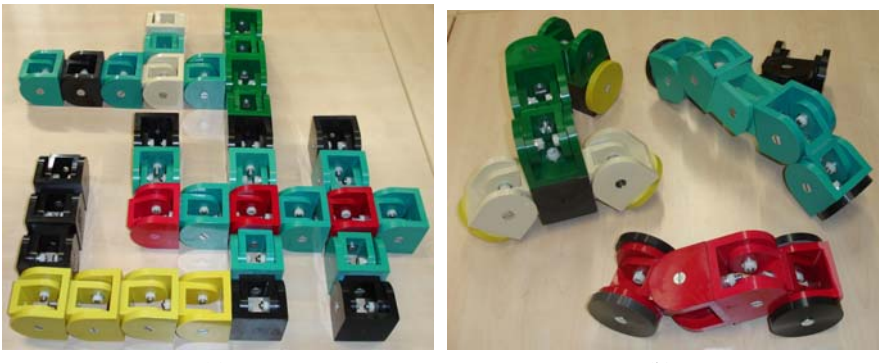


(d) (e) (f)

Fig. 4.45 Different topologies, produced by generators from Table 4.6.

Table 4.7 Types of structures and scalability (hom. – homogeneous, het. – heterogeneous).

Type of Structure	Scalability	Class
Caterpillar-like, hom.	n of segments $\underline{\underline{C}}^n_{sn}$, n of cross-caterpillars $Con^n()$	scalable
Dragon-like, hom.	n of marginal elements $Mar^n()$, n of segments $\underline{\underline{C}}^n_{sn}$	scalable
Centipede-like, hom.	n of marginal elements $Mar^n()$, n of leg's segments $\underline{\underline{C}}^n_{sn}$, n of cores in the body $Con^n()$	scalable
Planar grid (fungi-like), see Fig. 4.46(a), het.	n of modules	super-scalable
Wheeled, het. see Fig. 4.46(b), Fig. 2.6	n of modules, degree of heterogeneity	scalable



(a) (b)

Fig. 4.46 Examples of topologies from Table 4.7. **(a)** Planar "fungi-like" mesh, where modules are connected to each other only to share computational and energy resources, without any collective actuation; **(b)** Several example of wheeled organisms, where wheels are, for example, the active wheel from Sect. 2.1.6, see e.g. Fig. 2.6.

Basically, there are several ways to scale up and down the topologies: to scale the number/diversity of strong cores (this makes the topology structurally stable), typically this is the way of making n-Centipedes; to scale the number and diversity of segments (legs); to scale the number/diversity of marginal elements; finally to connect several organisms, however this can be applied mostly to planar structures, as shown in Fig. 4.46(a). Thus, during the self-assembling phase, robots can consider not only the choice of one pre-selected pattern, but also can scale up/down this pattern, see Sect. 4.5.4.

However, we face here the scalability problem, expressed by the fact that several topologies are not very scalable (i.e. they are not super-scalable, see e.g. (Constantinescu *et al.*, 2004)). There are several reasons why some structures are

not scalable, most important are appearing constraints Υ, which limit size and functionality. Below are listed several from such constraints:

N of available robots Υ^N	a topology can be self-assembled only when there are enough available robots;
Functional constraints Υ^F	imposed on the kinematics, for example specific degree of freedom to allow e.g. legged locomotion;
Weight of the modules Υ^W	the total weight of an organism is limited by capabilities of motors to drive this weight;
Structural constraints Υ^S	docking mechanisms allow only a specific static and dynamics stress, overstepping can destroy them. Scaling up structures should not overstress the docking elements.

Generally, the more functional requirements are imposed on the topology, the less scalable is the structure. For example, the fungi-like planar grid, shown in Fig. 4.46(a) is the best scalable topology, exactly because it does not have any collective actuation.

4.5.4 Morphogenesis as an Optimal Decision Problem

Morphogenesis or a topology generation is a distributed process, undertaken by all robots going to aggregate into the organism. It means that each robot module R_i has two independent decision processes about:

- **topology of a future organism** $\underline{\underline{C}}$, where all robots are represented by a placeholders R_i. This is a selection process, where the best topology is selected from a set of possible topologies. This also means that each robot module has its own topology matrix $\underline{\underline{C}}$, i.e. its own view on the topology of a future organism. The self-organization means in this case that all robots finally select the same topology without having a centralized decision process;
- **mapping** $ID_i \rightarrow R_i$**,** i.e. choosing a partner R_i for docking, i.e. the generation of $x_l^i \leftrightarrow x_k^j$ is independent of each other. This task is a constrained assignment problem, where a generation of the topology matrix underlies several requirements: firstly it should satisfy local constraints υ_i, secondly, each $x_l^i \leftrightarrow x_k^j$ pair has an associated local cost, and finally, the whole appearing topology has its own global costs.

Further we briefly introduce a generation of topologies and mapping $ID_i \rightarrow R_i$ leading to dynamic self-aggregation.

I. Generation of a topology. One of the key elements of this approach consists in a priori definition of topologies, which involves human and computational intelligence. We can imagine here three possible scenarios:

1. **All allowed topologies are in advance predefined patterns** ϕ_i**, without any perturbations of these patterns.** The dynamics in this case consists of two steps. Firstly, a set of predefined patterns ϕ_i (defined symbolically or by generators) is

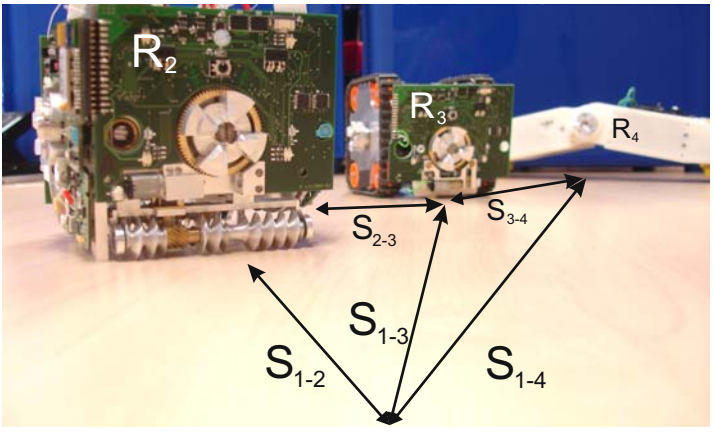


Fig. 4.47 Local R_1 robot's view: estimation of distances between robots.

compared with the current situation. For example, when other robots are already building some structure, this structure has a higher priority in the decision process. When there are no such structures, a robot compares current positions of known robots in order to decide which pattern is the most suitable for this configuration or which pattern is the most suitable for the given surface. Secondly, when a pattern is selected, a robot solves constrained assignment problem in order to determine a position in this pattern, where it goes to dock.

2. **Perturbation of any of the predefined patterns ϕ_i is allowed. This perturbation can be considered as a learning factor.** The dynamics in this case also consists of two steps. Firstly, a robot decides about the pattern ϕ_i, as in the first case. However, it is not constrained by this pattern, a robot can perturb the pattern ϕ_i by templates. This can happen only in two following cases. In the first case, the pattern, generated for n robots, already has n robots. The $n+1$ robot, joining to this organism, starts building a new scalability core. For example, when an organism with n robots has four legs, the $n+1$ robot can start building additional legs. In the second case, based on observation in the environment a robot can estimate a need of specific perturbation, e.g. to make legs longer to overstep some obstacles. However, the difficulty is that other robots may not know about this initiative and the whole pattern become desynchronized. Solution of this problem involves more communication between robots, as indicated in Sect. 4.5.5.
3. **The topologies are dynamically generated on demand by applying structural operators.** This is most complex case, because the generation should be performed also in distributed way and requires multiple synchronization steps. Moreover, the global costs, which are necessary for distributed self-assembling, should be also generated on-demand; this represents an unsolved issue so far. Therefore this case is not considered here (see Sect. 4.2 for a dynamic generation of topologies).

Before considering the second point of mapping $ID_i \rightarrow R_i$, we introduce the issues of local and global costs, which are necessary for optimization processes.

Local costs. Robots have different on-board capabilities to measure distances between robots, their orientation, relative rotation and other parameters (Kernbach *et al.*, 2009a). Moreover, as shown in Fig. 4.47, robots can measure distance not only to direct neighbors, but also to any visible object/robot. However, the further away the robots are, the less accurate is the measurement.

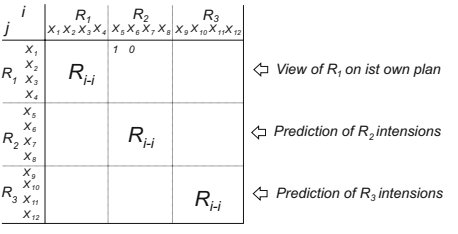


Fig. 4.48 Prediction of robot intentions through the topology matrix $\underline{\underline{C}}$.

The measured distance $S_{i,j}$ between R_i and R_j is one of the local costs for docking: the closer R_i and R_j are to each other, the "cheaper" is their docking $x_l^i \leftrightarrow x_k^j$. Other costs of $x_l^i \leftrightarrow x_k^j$ are the distance cost αS_{l-k}^{i-j}, rotation cost βS_{rot}^i, cost of "being hidden" S_{k-hid}^i, where α, β are coefficients. The cost S_{k-hid}^i is the price for the robot R^i of not-knowing the situation around R^i. For example, when the costs of connection between $x_1^1 \leftrightarrow x_1^2$ are $S_{1-1}^{1-2} = \alpha S_{1-2}^{1-2} + \beta S_{rot}^2 + S_{1-hid}^2$. More generally, local costs can include also any other factors, which determine a value of a particular connection. All local costs between all $x_l^i \leftrightarrow x_k^j$ are collected in the local costs matrix $\underline{\underline{S}}$.

Since morphogenesis is distributed and egocentric, the generations of $x_l^i \leftrightarrow x_k^j$ and $x_k^j \leftrightarrow x_l^i$ are asymmetric, i.e. from the viewpoint of the module R^i a cost of connection between R^i and R^j may be different then the connection between R^j and R^i from the view point of the module R^j. This leads to the effect, that the robot R^i knows precisely only its local costs, all other costs can be estimated only roughly. This situation is demonstrated in Fig. 4.48, where the first row represents an exact plan of the robot R_1, whereas all other rows are predictions of possible activities of other robots. When robots are close to each other and their local costs are similar, all topology matrices are expected to converge to each other.

Global costs. Issue of global costs represents a crucial point. In general, it means how good the whole organism fits the environment and can be expressed as velocity of motion, energy consumption, weight and any other global factor for a specific environment. Normally, it requires multiple tests with different configurations and is very expensive or even impossible for on-line estimation in real situations. Therefore the proposal is to use a set of different a-priori-tested topologies, whose global costs for different environments are known. For example, following the scenario 1 or 2 from the previous section, the efficiency of legged, wheeled or other topologies can be tested before experiments. During experiments, depending on availability of tools and sensed environment, robots have to agree in which configuration they should collectively work.

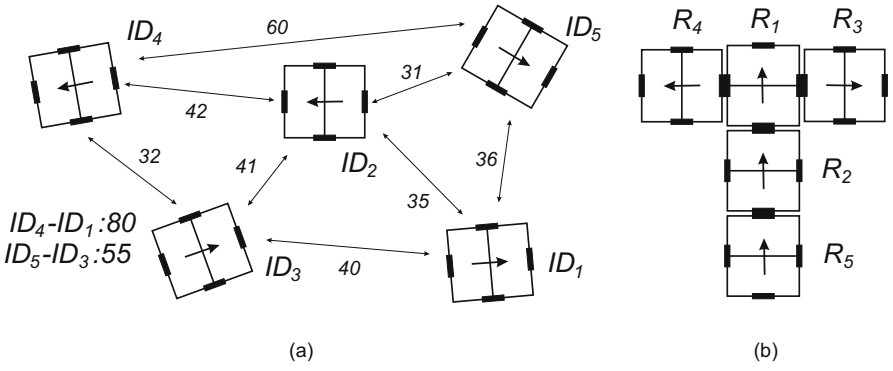


Fig. 4.49 Illustration of the constrained assignment problem. **(a)** Position and local costs of the robots, ID_i is a robot's number; **(b)** Topology of an organism, R_i is a place holder for a robot.

II. Mapping $ID_i \rightarrow R_i$. The generation of the mapping $ID_i \rightarrow R_i$ represents a classical assignment problem with one additional element: not only one docking element should be connected only one time (i.e. constraints (4.28)), but also these connections should be concentrated in specific parts of the adjacency table, which correspond to the selected topology. We illustrate this problem in Fig. 4.49(a) with a random placement of 5 robots, whereas Fig. 4.49(b) demonstrates a topology in which these robots should self-assemble. To make this consideration as simple as possible, we set local costs as only distances between robots (not between docking elements).

The problem, shown in Fig. 4.49, can be formulated as e.g. quadratic assignment problem (QAP), see e.g. (Loiola *et al.*, 2007) or as constraint-satisfaction problem (CSP) and constraint-optimization problem (COP), see e.g. (Kornienko *et al.*, 2004a). Since QAP is a NP hard problem, we will solve this in the CSP+COP way.

Classical assignment means that each $ID_i \rightarrow R_i$ is unique. However in our case, when for example $ID_1 \rightarrow R_1$, two additional robots should be connected to ID_1 (so that the costs of such connections are minimal). In the following table we display horizontally costs of all possible permutations between $ID_i \rightarrow R_j$ and vertically the connections $R_i \rightarrow R_j$ from the Fig. 4.49. The cost matrix takes the following form

$$\underline{\underline{S}}_{R_1} = \begin{array}{c|cccc|cccccc} & 1:2 & 1:3 & 1:4 & 1:5 & 2:3 & 2:4 & 2:5 & 3:4 & 3:5 & 4:5 \\ \hline 1:2 & 35 & 40 & 80 & 36 & 41 & 42 & 31 & 32 & 55 & 60 \\ 1:3 & 35 & 40 & 80 & 36 & 41 & 42 & 31 & 32 & 55 & 60 \\ 1:4 & 35 & 40 & 80 & 36 & 41 & 42 & 31 & 32 & 55 & 60 \\ 2:5 & 35 & 40 & 80 & 36 & 41 & 42 & 31 & 32 & 55 & 60 \end{array} \tag{4.36}$$

where additionally to the constraints (4.28), the assignment should satisfy

$$C_{ID_1 \rightarrow R_1} = \sum_{i}^{4} \sum_{j}^{4} S_{i,j} = 3. \tag{4.37}$$

The constraint (4.37) is a degree of connectivity for the main core element R_1. Solving the assignment problem, taking into account (4.37), we receive the following assignment matrix

	1:2	1:3	1:4	1:5	2:3	2:4	2:5	3:4	3:5	4:5
1:2	1	0	0	0	0	0	0	0	0	0
1:3	0	1	0	0	0	0	0	0	0	0
1:4	0	0	0	0	0	0	1	0	0	0
2:5	0	0	1	0	0	0	0	0	0	0

(4.38)

where the associated costs are $S_{R_1} = 186$.

When, for example, solving (4.36) without considering (4.37), we obtain

	1:2	1:3	1:4	1:5	2:3	2:4	2:5	3:4	3:5	4:5
1:2	0	0	0	0	0	0	1	0	0	0
1:3	0	0	0	0	0	0	0	1	0	0
1:4	1	0	0	0	0	0	0	0	0	0
2:5	0	1	0	0	0	0	0	0	0	0

(4.39)

which has associated costs 138 (lower than in (4.38)), however it does not represent the dragon-like topology from Fig. 4.49(b).

Now we should repeat the step (4.36), however for $ID_2 \rightarrow R_1$, i.e for the following cost matrix with the same constraint (4.37) which creates the following assignment

$\underline{S}_{R_2} =$

	2:1	2:3	2:4	2:5	1:3	1:4	1:5	3:4	3:5	4:5
1:2	35	41	42	31	40	80	36	32	55	60
1:3	35	41	42	31	40	80	36	32	55	60
1:4	35	41	42	31	40	80	36	32	55	60
2:5	35	41	42	31	40	80	36	32	55	60

(4.40)

	2:1	2:3	2:4	2:5	1:3	1:4	1:5	3:4	3:5	4:5
1:2	1	0	0	0	0	0	0	0	0	0
1:3	0	1	0	0	0	0	0	0	0	0
1:4	0	0	0	1	0	0	0	0	0	0
2:5	0	0	0	0	0	0	0	1	0	0

(4.41)

with $S_{R_2} = 139$ as costs. Proceed further for $ID_3 \rightarrow R_1$, $ID_4 \rightarrow R_1$ and $ID_5 \rightarrow R_1$ we can compose the following costs table

S_{R_1}	S_{R_2}	S_{R_3}	S_{R_4}	S_{R_5}
186	139	148	165	157

, (4.42)

where we immediately ascertain that the assignment (4.41) has the minimal costs. In other words, when the robot ID_2 is used as a central core element, this will create the smallest movement in the whole group, as it is well visible from the Fig. 4.49. The considered example does not involve docking elements (i.e. it is not specified from which side robots should dock to each other) and is intended only to illustrate the constrained assignment problem.

In the example above, we imposed only one constraint (4.37) on the elements with the maximal degree of connectivity. More generally, we should consider the degree of connectivity for all involved elements R_i and to impose corresponding constants on the cost matrix $\underline{\underline{S}}$.

Revising the approach based on (4.36)-(4.42), we can make three remarks:

- The (4.38) contains direct connections between ID_i, e.g. $ID_1 \rightarrow ID_2$, $ID_1 \rightarrow ID_3$, $ID_1 \rightarrow ID_4$, $ID_2 \rightarrow ID_5$, moreover they are related to corresponding connections between R_i, for which all kinematic properties are known – this represents an advantage of the representation style (4.38).
- The list of all possible permutations of $ID_i \rightarrow ID_j$ may be large.
- The whole approach consists of considering all ID_i as candidates for R_1 connectivity degree 3), then for R_2 (connectivity degree 2) and then all other elements.

The last point opens the way of stepwise solving a series of the low-dimensional constrained assignment problems, which can have the following form:

1. All topologies have a list of associated constraints Υ: total N of robots, degrees of connectivity, requirement on heterogeneous modules and so no. Moreover, each topology has a list of global costs: consumed energy, velocity of motion on a normal surface, geometry of concave and convex obstacle treatable with this topology or a possible geometry of docking elements.
2. Before start self-assembling, robots check whether N of available robots match the set of possible topologies. For instance, when there are topologies requiring $\{15, 17, 22, 25\}$ robots and there is only 20 available robots, they can self-assemble only into first two topologies.
3. Several topologies require tools or specialized robots. Robots should check availability of these specialized robots and correspondingly limit the set of possible topologies.
4. Self-assembling starts from the core element with the highest degree of connectivity. When such a core element is already built, robots consider the next element with the lower degree of connectivity and so on, until the whole structure is assembled.
5. When the selected topology is already partially assembled, and more free robots arrived to the assembling place, robots can decide instead of disassembling and new assembling to create a new core in the already assembled structure. Prerequisite is that the topology can be scaled up in the number of cores.

The considered approach involves the costs matrix $\underline{\underline{S}}$ where all costs are globally observable. In real situation this is not feasible without the use of a central element. In the next section we demonstrate conditions when even a partially observable matrix $\underline{\underline{S}}$ leads to the same assignment as a full costs matrix. This feature can underlie the self-organized morphogenesis, performed in a completely distributed way.

4.5.5 Self-organized Morphogenesis

As already mentioned, self-organization is a process, when ordered collective behavior appears without a central coordination or control. The approach, represented

in Sect. 4.5.4, assumes that the cost matrix (4.36) collects global information about all robots, i.e. requires a central instance. In this section we demonstrate, that local multiple views, as shown in Fig. 4.47, even when some costs are not observable, can lead to the same or to similar assignment. In other words, this approach represents a synchronization mechanism, underlying a self-organizing solution.

Firstly, we formalize the algorithm from the previous section. It is assumed, that all topologies are accessible through the index i in ϕ_i, all constraints are collected in Υ in two following groups: structural constraints (e.g. υ_i^N - imposed on the number of robots, υ_i^S - specialization of robots), and environmental constraints υ_i^{Env} imposed on the functionality of an organism (e.g. the length of legs or already built cores).

Algorithm 3. Morphogenetic algorithm

```
1  Communicate(with everybody) → to collect Υ^Env and Φ
   // Step 1. Removing such φ_i which does not satisfy Υ
2  forall elements of φ do
3  |  forall elements of Υ do
4  |  |  if constraints υ_j != satisfied then remove φ_i from Φ
5  |  end
6  end
   // Step 2. Decide/Generate a final topology
7  Select φ_i with minimal global costs
   // Step 3. Solve iterative assignment problem
8  Create local cost matrix S for all observable ID_i → ID_j
9  forall R_i with decreasing degree of connectivity do
10 |  Solve particular assignment problems
11 end
   // Step 4. Self-assembling phase
12 Communicate(with the selected neighbor) → to confirm docking intention
13 if confirmation == successful then Dock to the selected partner else goto 1
```

Now we comment this algorithm. First of all, robots should create common knowledge about the current situation, namely the number of robots N, the environmental constraints Υ^{Env} (which include also information about already built cores) and known patterns Φ. Such a collection of common information can be achieved based on distributed (e.g. token-based) algorithms and is done in the line 1. After the robot removes such patterns, which do not satisfy constraints (lines 2-6). From the remaining patterns, only one (or one core) should be selected by minimal local costs (line 7). When some core (or even part of an organism) is already built, corresponding pattern will be automatically selected here (because only this pattern satisfies the environmental constraints). For all required connection $R_i \rightarrow R_j$, a robot measures local costs for the corresponding $ID_i \rightarrow ID_j$, creates the local cost matrix $\underline{\underline{S}}$ (line 8) and iteratively solves assignment problems (lines 9-11) as described in the previous section. Finally, a robot contacts the selected partner and performs a

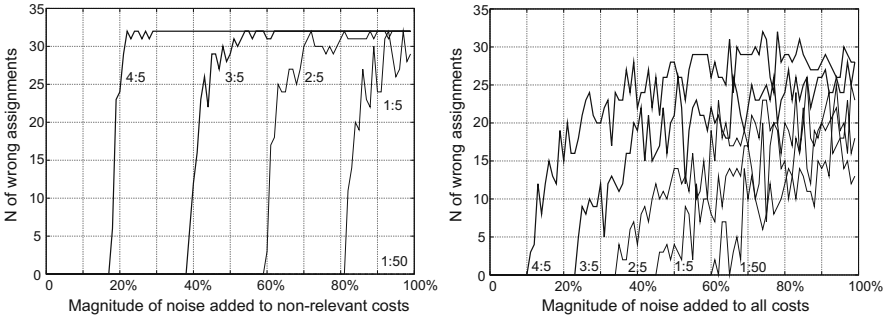


Fig. 4.50 Solution of 32×32 assignment problem with noise added to the cost matrix. Shown is the number of wrong assignments related to the original assignment without noise. Numbers x:x (e.g. 1:50) demonstrate a relation of "low" costs leading to "1" in the assignment to those costs which lead to "0" in the assignments, for example 1:50 means that "0-leading costs" are 50 times larger than "1-leading costs". **(a)** Five cases (1:50, 1:5, 2:5, 3:5 and 4:5) of assignments where the noise is added only to "0-leading costs"; **(b)** The same five cases of assignments, where the noise is added to all costs.

docking approach. When the selected partner refuses docking, a robot updates the list of constraints and repeats the whole procedure (line 12).

Due to the same sets of constraints and initial topologies, all robots have the same selected pattern ϕ_i. The difference between robots appears first at the line 8, when generating a local cost matrix. Basically there are two sources of such a difference: poor measurement accuracy for large distances and non-visible situation for a particular robot. The distributed generation of the cost matrices leads to deviations in those matrices, i.e. the estimated costs will differ from each other and from real "original" cost matrix.

We investigate this process in more detail. Let us consider the matrix (4.40) (the square part separated by the line), for 32×32 assignments (8 robots with 4 docking elements, the notation is same as suggested in Sect. 4.5.1). Each cost matrix contains such a combination of costs, which leads later to "1" in assignments. We call these costs as "1-leading costs", whereas other are "0-leading costs". There are five different relations between these costs, shown in Fig. 4.50 as 1:50, 1:5, 2:5, 3:5 and 4:5. Each element of the cost matrix is perturbed by the noise value $\pm k$ from $k = 0$ (0%) until $k = max\ cost$ (100%). We also consider two cases, when noise is added only to "0-leading cost" and to all costs. All these results are shown in Fig. 4.50.

It is clearly visible that the level of noise and the relation between 1- and 0-leading costs are contrary factors: the higher is this relation, the lower should be the noise level. Considering realistic values of 3:5, 4:5, we see that correct assignments can be achieved for the noise level of 35% and 15% correspondingly for the first case (Fig. 4.50(a)) and 20% and 10% correspondingly for the second case (Fig. 4.50(b)). Obviously, that within these boundaries, distributed generation of the costs matrices will lead to the same assignment, i.e. this process is self-synchronizing without a need of central instances. When the level of noise (i.e. accuracy of measurements) is

outside of these boundaries, robots should communicate in order to improve quality of the cost matrix $\underline{\underline{S}}$ – it is enough to transmit only own well-estimated costs to other robots.

4.5.6 Collective Memory and Further Points

There are several further and concluding points to this section. First, it needs to consider the problem of environmental constraints Υ^{Env}, which generally speaking, may be unknown at the begin of self-assembling. Even exchanging of Υ^{Env} between robots does not solve this problem, because some constraints should be first encountered by exploring the environment. Therefore the exportation and adaptation loop is a part of the morphogenetic process. Below in Algorithm 4 we demonstrate a general view on morphogenetic processes in artificial organisms.

The iterative process of exploring environment and changing a topology represents a basic feature of structurally adaptive systems. There are two points here, which need further consideration. Firstly, encountering a new environmental constraint υ_i^{Env} can basically change the solution space for CSP+COP, this new υ_i^{Env} can make the problem unsolvable. In other words, it may happen, that all topologies Φ, even their perturbations and scaling, do not satisfy Υ^{Env}. In this case, a new topology ϕ_i should be generated in *trail-and-error* manner. Such a on-line generation of topologies is considered in other sections of this book. Secondly, achieving the task, robots collect experience about a structure of environment and about possible ways of solving problems in this environment. Related to topologies, there are two important sets Υ^{Env} and Φ, which represent environment and solutions. This experience is distributed among all robots, we can associate it with a collective memory of the organism. An obvious question is what will happen with this memory after disassembling ?

Algorithm 4. General morphogenetic process

```
   // Step 1. Initialization
1  Communicate(with everybody) → to spread initial common knowledge (e.g. tasks)
2  Communicate(with everybody) → to collect Υ^Env and Φ
3  Collectively choice a topology φ_i and Self-assemble into φ_i
   // Step 2. Achieving the task
4  while task not achieved do
5      if new constraint in υ_i^Env encountered then
6          Update Υ^Env, Disassemble from old φ_i
7          Collectively choice a new topology φ_i, Self-assemble into this new φ_i
8      end
9  end
   // Step 3. Save collective experience & disassemble
10 Save updated Υ^Env and updated Φ
11 Disassemble
```

There are several ideas about such a collective memory. We should remark, that each module can act within different organisms and with different tasks. Individually saved information about Υ^{Env} and Φ may be inconsistent and even have a conflicting character. Therefore it may be important to have a kind of central "wise robot" or "oracle", which collects all information and stores it in a consistent way. This may lead to social specialization, when robots have such roles, which do not lead directly to solution of a task, but maintain a collectiveness of the group (i.e. this decreases collective energetic efficiency). Another solution consists in storing all organism's relevant data locally in each robot (for example, using large flash memory), i.e. each robot can reconstruct the whole organism (e.g. information, stored in DNA). The question about effectiveness of social and genetic way of representing collective memory is open and required further research.

In summary, we have introduced representations of topologies by matrices, in symbolic way and by a generator. This provides different possibilities to handle on-line and off-line morphogenesis, on-line adaptation and scalability. The constrained assignment in the form of CSP+COP was introduced to on-line morphogenesis. Linear optimization, performed in the distributed way, can underlie the self-organized self-assembling process, when the accuracy of local measurement lies within the indicated noise level.

We point out that using optimization of off-line pre-evolved topologies instead of on-line evolving has an advantage of faster and more efficient morphogenesis. However, we also indicated that for unknown local costs in $\underline{\underline{S}}$ or for unsolvable Υ^{Env} and Φ, on-line evolving loop is required. In other words, for such situations, which cannot be a priori evaluated, robots should perform on-line and on-board morphogenesis.

4.6 Kinematics and Dynamics for Robot Organisms

Eugen Meister

From a mechanical perspective, a self-reconfigurable multi-robot organism is a set of connected joints and links which are determined by kinematic parameters (joint angles, accelerations, acting forces etc.). Traditionally, in order to analyze kinematics and dynamics of a rigid body system, several synthetical and analytical approaches such as Newton, Euler, Hamilton or Lagrangian formulations are dominating. In the area of self-reconfigurable robotics, we face the problem that a morphology of the robot after configuration is not predefined at all and hence a formulation of kinematics and dynamics with these standard approaches is possible only in a limited way. The kinematical model cannot be fixed in advance due to different tasks which require environment-dependent topology. Those autonomous systems should first be able to recognize all interconnections between all links in a body and to build a map (graph, adjacency matrix etc.) of the whole topology. In SYMBRION/REPLICATOR we primarily consider all possible shapes of such multi-robot organisms, however after performing some analysis in Sect. 4.5 of

reasonable topologies, we came to the conclusion that unsymmetrical structures require more complex description. Classification of all reasonable morphologies into "unbranched" (snake, wheel etc.), "branched" (snake-like, spider-like etc.), "small-scale" or "large-scale" multi-robot organisms is useful for assignment between tasks and topologies (Chen, 1994). This classification can simplify, however not solve the problem of how to generate a model for kinematics and dynamics automatically. Here adaptive methods are required, which are capable of automatical formulation and reformulation of required models for kinematics and dynamics. Therefore, the first challenge is to create a scalable model which is autonomously able to change and to extend/reduce parameters, depending on the structure of the robot organism. Another big challenge is the reduction of complexity of such a model, so that organism's dynamics can be calculated and controlled in real time. In this section, several approaches are discussed, analyzed and compared in order to find a suitable solution for the robot platforms.

Recently, a collection of efficient algorithms for analysis, syntheses and control of kinematics and dynamics has been developed. However, the performance of such dynamic algorithms often does not only depend on the algorithm itself, but also on the chosen representation techniques. The big classification can be done by separating the algorithms either by using absolute or relative coordinates. Absolute coordinates use a fixed frame as the reference frame. In contrast to absolute coordinates, relative coordinates take local link frames as their reference. Representation in absolute coordinates is often used for analyzing and control of rigid body systems, especially in industrial applications. Several advantages such as uniformity or easy representation of constraints make this approach dominant for using in many commercial packages. However, every interaction between bodies in this representation technique has to be represented by a large set of constraint equations and the dynamic equations get difficult to solve. By contrast, the number of constraint equations can be minimized by using relative coordinate approaches and these approaches nowadays become more popular, especially in the field of self-reconfigurable, modular and mobile robotics.

The performance and efficiency of dynamic algorithms depend also on their mathematical notation. The 3D vector notation in Euclidean space is one of the traditional methods for the formulation of rigid body kinematics and dynamics, however in the last few years, the so called "spatial notation" built on the 6D vector notations, became widely accepted in many different solutions for kinematics and dynamics problems. The 6D vector notation, dated back to 1866 first introduced by Pluecker and nowadays representation in Pluecker coordinates, is often used because it leads to efficient computer implementation (Selig, 2004). In order to express the dynamics of a rigid body with six degrees of freedom in 3D space, we usually need two equations: the first expression for formulating the relationship between forces and linear accelerations, and the second equation in order to relate moments and angular accelerations. The representation in spacial notation combines these two equations into a single one. The idea behind this notation is to merge linear and angular velocities into a common vector as well as to merge forces and moments in the same manner. This notation has additional advantages and simplifications:

1. The equations of motion are simplified;
2. Algebraic calculations are reduced at least by factor four, compared to traditional 3D vector notation;
3. Writing computer code is simplified, since code gets shorter and easier to understand.

Several rules and further advantages of this notation are summarized in (Featherstone, 2008).

The spatial notation in Pluecker coordinates can be described by a line, linear and angular magnitudes are closely related to the screw theory (Ball, 1900). One powerful extension applied to this theory was introduces by (Murray *et al.*, 1994), where all general quantities (velocities, accelerations, forces etc.) are expressed in terms of linear operators on $se(3)$, which is related to the Lie Algebra of Special Euclidean Group $SE(3)$. There exist also several implementations for serial and branched multi-rigid body systems based on this extension, introduced in (Chen, 1994).

In the following sections, we explain several representation and modeling techniques for self-reconfigurable robots and show how they can be applied to the described multi-robot organisms. The results are based on a careful analysis and provide a basis for developing new modeling concepts.

4.6.1 Modeling of Multi-robot Organisms

In the last view years, several new modeling techniques for reconfigurable rigid body systems appear. In the industrial applications, reconfigurability has an advantage because the same robots are able to perform different tasks only by changing their shapes and functionality. Since in most industrial applications, the number of possible structures is limited, it is enough to determine only a few models for kinematics and dynamics and to switch between them if a new configuration is required. In this case, the level of adaptability is well-arranged. However, if we can expect a scalable number of possible configurations, adaptive modelling mechanisms are required, which are able to handle the complexity in a smart way without using a predefined set of possible configuration models for kinematics and dynamics.

Traditionally, the analysis for multi-body systems is done by using algebraic formulations, however the complexity increases rapidly with the number of degrees of freedom (DOF) or with the number of redundant modules in a robot. In other words, the methods are not scalable.

Opposed to these traditional formulations, there exists another possibility to threat the problem in a geometrical way and with analytical set of tools. One of the most promising approaches is based on the screw theory for body kinematics. It is often combined with mathematical methods based of Lie groups and Lie algebras (Chen, 1999). This geometrical formulation has several advantages:

- Screw theory provides the geometrical description of rigid motion;
- To distinguish between prismatic, revolute or other kind of joints is not required opposed to traditional Denavit and Hartenberg (DH) parameters approach (Denavit & Hartenberg, 1995);

- No suffer from numerical ill conditions like those in DH parameters;
- Allow global description of rigid body motion. The problem of singularities disappears due to using local coordinates.

Screw theory is an old theory, developed more than 100 years ago by Sir Robert Stawell Ball and unfortunately falls into oblivion during the last 80 years. The principle is based on the screw motion which is a combination of translation along the screw axis and a rotation around it. Hence, any general motion in 3D can be represented by a screwing motion described in the Chasles's Theorem (Fig. 4.51). In screw theory terminology, such a motion is called a "twist" and the forces acting along a screwing line are called "wrenches".

In Fig. 4.51, to explain used quantities: "$" symbolize the instantaneous screw axis, ω represent the angular velocity measured in radian/s, h is a pitch of the screw motion and v is the velocity vector of the point A.

4.6.1.1 Automatic Model Generation

One of the biggest challenges in modular reconfigurable robotics is to achieve a high degree of adaptivity with regard to the model of the system. In order to avoid manual human intervention, a multi-robot organism has to adapt their models for kinematics and dynamics according to the tasks which should be fulfilled. Automatically generating a model means, autonomously recognize the topology of the organism and based on it derive a closed-form equation of motions. In order to start automation process, a starting point is required and therefore a "base module" should be chosen by the system itself. In the starting phase of aggregation it is obviously that the first module that gets the desired task allocation starts with building the organism and hence becomes the base module. It is not necessary that the first module stay as a

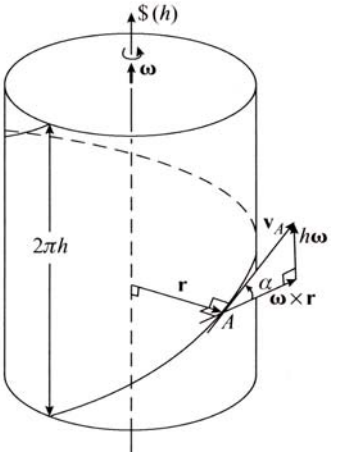
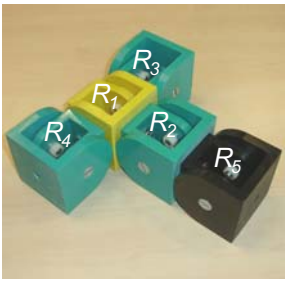
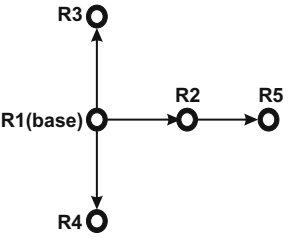


Fig. 4.51 Principal of screw displacement which is a combination of translation along the screw axis and a rotation around it (Davidson & Hunt, 2004).



(a)



(b)

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(c)

Fig. 4.52 Dragon-like organism structure. (a) 3D model, (b) Directed graph representation, (c) Accessibility Matrix.

base for the whole time period, the role of a base module can be assigned to every module during run time. Since aggregation of modules to an organism is better performed in planar configuration when robots are on the ground (see Sect. 4.5), negotiation for selecting the base frame can happen during the docking phase. After the base frame has been chosen, forward kinematics for each branch outgoing from the base element can be calculated as a next required step. In (Chen, 1999), an Accessibility Matrix based on graph representation has been introduced which provides connectivity information from the base module to every pendant module in each branch. In this matrix elements are equal to one if there is a directional/bidirectional path between modules, otherwise the elements are zero. This matrix has for example the following form if we look to dragon-like organism in Fig. 4.52.

However, what is still missing is the information about the docked sides of each robot. This information can be obtained from the topological matrix $\underline{\underline{C}}$ introduced in Sect. 4.5. Combining this information from both matrices the organism is able to start calculating the forward kinematics.

4.6.1.2 Forward Kinematics

Calculation of forward kinematics is one of the first steps required to get a model of a multi-robot organisms. Based on dyad kinematics (a pair of connected links in a kinematic chain), with combination of topology matrix, we are able to calculate forward kinematics for a branched/unbranched robot recursively. The core of forward kinematics formulation in screw coordinates is based on matrix-exponential formula, a mapping onto $SE(3)$

$$map\ exp:\ se(3) \longmapsto SE(3) \tag{4.43}$$

and usually defined as

$$e^{\hat{\zeta}\alpha} = I + \hat{\zeta}\alpha + \frac{\hat{\zeta}\alpha^2}{2!} + \frac{\hat{\zeta}\alpha^3}{3!} + \ldots, \tag{4.44}$$

where $e^{\zeta\alpha} \in SE(3)$ map a point from an initial to the final pose. This formulation is usually used for open-chain rigid body systems, but can be extended to threat as well closed-loop type of robots, however this is not considered in this book.

Forward kinematics for each branch can be calculated recursively starting from the base module. For example, in a multi-legged robot organism, the base module can be one in the middle of the structure. In order to give the model representation in screw coordinates, determining of twists vectors are necessary. Twists is a skew-symmetric matrix representation of the joint axis, it represents the velocity of a rigid body motion geometrically.

$$\zeta_j = (v_j, \omega_j) \tag{4.45}$$

where $\omega_j, v_j \in \mathbb{R}^3$ and are the linear and the angular velocities expressed in spatial notation. Starting from initial configuration in a plane $g_{st}(0)$, the forward kinematics map $g_{st}(\alpha)$ for open chain manipulators can then be obtained by using the Product of Exponential method (POE) made popular by Roger Ware Brockett (Brockett, 1984).

$$g_{st}(\alpha) = e^{\hat{\zeta}_1\alpha_1} e^{\hat{\zeta}_2\alpha_2} \ldots e^{\hat{\zeta}_n\alpha_n} g_{st}(0), \tag{4.46}$$

where α is a measure angle for rotation about axis, $\zeta_1..\zeta_n$ are the corresponding twists numbered sequentially starting from base frame. The equation can be considered as a Product of Exponentials of twists. The fact that we need only two coordinate frames, the base frame and the final frame in a chain combined with geometrical significance of twists make this approach to a favorably in comparison to traditional approaches.

Let us consider a simple set of modules forming a snake-like robot. First of all, it is necessary to determine all interconnections. In Sect. 4.5, a topological matrix $\underline{\underline{C}}$ was introduced, which give us the information about how robots are connected with each other. To determine twists from this matrix, we should be able at least to read out the angular velocities direction vectors. Therefore we fist need to fix the directions according to the face site of the module

$$\Longrightarrow \hat{=} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \Longleftarrow \hat{=} \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \qquad \Uparrow \hat{=} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \Downarrow \hat{=} \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$$

and as a next step apply this definition to our robot.

Starting with the base module and following for each module in the organism, the corresponding twists and the forward kinematics using the Eq. (4.46) can be calculated recursively for all branches in the organism.

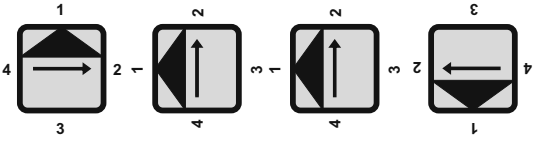


Fig. 4.53 Principal of connected modular robots and their twists directions.

4.6.1.3 The Jacobian

The forward kinematics (Eq. 4.46) provide information about the positions represented by the joint angles α_i, however of a big interest are often the velocities, especially of the last module in the chain, ussually called a tool frame. The relationship between the velocities of joint motors and the velocity of the tool frame is often named as "differential kinematics". Traditionally, going from position kinematics to the differential kinematics is just a matter of taking the time derivate, however this hold only if the Jacobian is a linear map. Choosing coordinates for $SE(3)$, to obtain the Jacobian is not that simple anymore and this formulation holds only locally. In order to correct this problem, the Jacobian can be written in terms of twists. Consequently, the instantaneous spatial velocity of the tool frame can be represented as follows:

$$\hat{V_{st}} = \dot{g}_{st}(\alpha) g_{st}^{-1}(\alpha). \tag{4.47}$$

Applying the chain rule and writing in twist coordinates we obtain

$$\hat{V_{st}} = J_{st}(\alpha)\dot{\alpha}, \tag{4.48}$$

where

$$J_{st}(\alpha) = \left((\frac{\partial g_{st}}{\partial \alpha_1}) {g_{st}}^{-1}{}^{\vee} \ \ldots \ (\frac{\partial g_{st}}{\partial \alpha_1}) {g_{st}}^{-1}{}^{\vee} \right) \tag{4.49}$$

is called the spatial manipulator Jacobian. The operator $\vee$ (vee) extract 6-dimensional vector which parameterizes a twist. Using the Eq. (4.46) for the forward kinematics we derive an elegant formula for J_st:

$$J_{st}(\alpha) = \left(\zeta_1 \ \ \zeta_2^{'} \ \ldots \ \zeta_n^{'} \right) \tag{4.50}$$

where

$$\zeta_i^{'} = Ad_{\zeta_i}, \tag{4.51}$$

where Ad is the adjoint transformation associated with ζ_i. This relationship between joint velocity and the tool velocity allow us to move this open chain from one configuration to another without calculating the inverse kinematics:

$$\dot{\alpha}(t) = (J_{st}(\alpha))^{-1} V_{st}(t), \tag{4.52}$$

If we know V_{st}, we need to integrate this differential equation over the time interval $[0, T]$ along the workspace path.

4.6.2 Inverse Kinematics

Given a desired configuration or trajectory for the last links (tool) in the chains of robot modules, the goal for the inverse kinematics is to find joint angles for all joints to achieve that configuration. Traditionally, inverse kinematics solutions are separated in two classes: closed-form solution and numerical solutions. There exist

different methods for solving inverse kinematic problem, however all of them are faced with similar problems like:

- Obtain a close-form solution;
- Reduction of complexity;
- Reduction of number of possible solution;
- Performance of the algorithm under real-time conditions.

To find a closed-form solution if often desired because it gives an efficient calculation but cannot always be obtained. In such cases, there is still a possibility to solve the problem numerically for example applying Newton Raphson's Iteration Method. Considering formulations based on screw calculus described in sections above, there exist another possibility to solve the inverse kinematics, namely with the geometric algorithm based on the POE formula for the forward kinematics map. In this method, the inverse kinematics problem is reduced into set of subproblems which are geometrically meaningful and numerically stable. To explain all possibilities and all tricks how to reduce the inverse kinematics problem into a set of subproblems would go beyond the scope of this book. Therefore, detailed explanations can be found in (Paden, 1986), (Kahan, 1983).

In a branched multi-robot organism, the calculation of inverse kinematics in order to reach a desired position of the tool in every branch shall perform simultaneously for all branches since branches may not be independent driven and share modules with each other. Therefore, the inverse kinematics for such a multi-robot organism is the calculation of a set of joint angles that will place every end-link to its desired pose simultaneously. For this reason, the kinematic equations should be combined to a single equation for example of the form:

$$T = J\, d\alpha, \tag{4.53}$$

where in spacial coordinates $T \in \mathbb{R}^{6n\times 1}$ is a pose vector and $J \in \mathbb{R}^{6m\times 6n}$ is a generalized body manipulator Jacobian. The dimensions of T and J depends on how many modules are in each branch and how many branches does the organism have. Looking to this big matrix which is growing proportionally with the number of modules the question arises if this method is scalable or not. In the previous section we learned that the Jacobian grow with the number of modules however the representation in twist coordinates and $SE(3)$ give us the possibility to solve the problem in elegant way and without high calculation costs.

4.6.3 Dynamics

The biggest challenge in order to control a robot, especially a multi-rigid body system with many degrees of freedom, is to find a solution for dynamics that is usually described by ordinary differential equations. These equations are often called motion equations because it describe the movement of the robot in response to actuator forces. There exist several of standard formulation methods for motion equations, however, two of them are mostly preferred: the Lagrangian and the Newton-Euler

equations. Both methods create similar sets of equations however the derivation of the equations has different backgrounds. Lagrangian method, often of complexity $O(N^4)$, relies on energy properties of mechanical systems and give a general method for deriving the equations of motion. However it cannot be directly used as long as the configurations space is parametrized by a subset of R^n, where n is the number of DOFs in a multi-body system. In order to apply this method for rigid bodies with configuration in $SE(3)$ like described in sections above, we need to choose local parameterization. Newton-Euler equations, often of a complexity of $O(N)$, are probably the most frequently used approach in the literature to formulate motion equations. Newton-Euler equations describe the dynamics of a multi-body system in terms of forces and torques applied to the object. Looking for computer implementation techniques, the recursive Newton-Euler method is the most widely spread dynamic algorithm and has been proved by many researchers like (Luh & Paul, 1980), (Orin & Hartoch, 1979) or (Stepanenko & M., 1976) to be more efficient, recursive and to adapt to be capable of real-time conditions. In general, looking to different dynamic algorithms we must distinguish the purpose it is to be used. In (Featherstone, 2008), algorithms for dynamics have been classified in two major classes:

- Forward Dynamics (FD);
- Inverse Dynamics (ID).

In forward dynamics, an acceleration response of a rigid body to applied forces has to be calculated. This technique is mostly used in simulations. In contrast to forward dynamic, the inverse dynamic performs calculation of the force that must be applied to a rigid body in order to produce desired acceleration response. This technique can be used for control design of the system, trajectory planning etc. Using either the Lagrangian or the Newton-Euler method we obtain a closed form motion equations of the form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau, \tag{4.54}$$

where q, $\dot{q}$ and $\ddot{q}$ variables represent position, velocity and acceleration, $M(q)$ is a mass matrix, $C(q,\ \dot{q})$ is a Centrifugal and Coriolis accelerations and $N(q)$ represents gravitational and external forces. Due to the elegant representation technique mentioned in the sections above, our intension is to represent all quantities in terms of twists and wrenches. Wrenches are force/moment pairs contain forces and moments usually applied at the center of mass. The values of wrenches depend on the coordinate frame in which they are represented:

$$F = \begin{pmatrix} f \\ \tau \end{pmatrix} \tag{4.55}$$

where $f \in \mathbb{R}^3$ is a linear force component, $\tau \in \mathbb{R}^3$ represents a rotational component and F is $\in \mathbb{R}^6$. Transformation of wrenches or twists can be performed by using Adjoint transformation presented above,

$$F_a = Ad^T_{g_{ba}} F_b, \tag{4.56}$$

where forces acting on the body coordinate frame B are written with respect to coordinate frame A. In spatial representation, this is equivalent as if the coordinate frame A were attached to the object. For a branching type of robot, the recursive Newton-Euler algorithm provides the best opportunity for fast calculation of kinematics and dynamics. This algorithm can be separated into two cycles. The first calculation propagate simultaneously from the chosen base module of the multi-robot organism to all branches in order to calculate general velocities and accelerations. In order to produce these accelerations, general forces have to be calculated and this is performed in backwards manner from the end modules in each branch to the base module. Finally the results for all branches are summed up. This strategy have been applied in different works (Chen, 1999), (Featherstone, 2008).

In (Featherstone, 2008), detailed considerations about the implementation issues, complexity etc. can be directly compared with the complexity we expect in calculations for robots. Several different algorithms for Forward Dynamics (Inertia Matrix Method, Propagation Method, The Articulated-Body Algorithm), Inverse Dynamics and even some Hybrid Dynamics (HD) algorithms (Articulated- Body Hybrid Dynamics, Floating-Base Forward/Inverse Dynamics) are introduced and compared with each other. We can use these algorithms in our projects and our intension is to profit from the experiences made in the past and additionally to extend these algorithms with additional features and use this knowledge to develop our own methods adapted to our platforms. As we can see in (Chen, 1994), the technique of using the Lie Algebra bring some additional advantages and allow an elegant representation and calculation techniques. Our first goal is to combine these two research areas and try to profit from both, on the one hand make use of effective dynamic algorithms and on the other hand reduce the calculation complexity additionally by using spacial Lie Algebra notation.

4.6.4 Computational Analysis

Among many different representation techniques, two have been emerged and found a particular popularity in the robotics community. One of traditional representation is the homogeneous transformation, based on 4×4 real matrices of homogeneous coordinates. Another representation discussed in sections above is a quaternion/ vector pair representation based on unit quaternions. In (Funda & Pual, 1990), four mathematical formalisms and therefore four distinct representations of a spatial screw displacement (dual orthogonal 3×3 matrix, dual unit quaternion, dual special unitary 2×2 matrix and dual Pauli spin matrices) are proposed. They are often used to describe spatial transformation (translation, rotation etc.) for lines in space. They have been analyzed regarding computational complexity and costs in terms of the required number of CPU cycles. These four representations have been analyzed in terms of their computational behavior in performing two basic operations:

1. General spacial screw displacement of a line;
2. Composition of two general spatial screw displacement operators.

All test algorithms in that paper have been implemented in sequential, as well as in parallel way. The paper concludes that the dual unit quaternions are the most efficient representation of screw displacement of lines (joint axes) in space. Since the computational capabilities of a multi-robot organism are quite limited, this analysis is important in order to decide about the used method in the concept development. Additional criterion contributing to the popularity of screw calculus is the fact that this can be applied not only for kinematics but also for statistics, trajectory planning or dynamics of spatial linkages problems. These can be formulated and analyzed in a uniform and consistent fashion.

4.6.5 Conclusion

Before we are able to control the multi-robot organisms during a locomotive phase, we need a model of kinematics and dynamics. In this section we described a possible alternative usage of a spatial notation based on the screw calculus. A combination of algorithms developed in (Featherstone, 2008) and in (Chen, 1999) is suggested. Both approaches describe the model in spatial notation, however with different focuses. The goal followed in (Featherstone, 2008) was to develop efficient dynamic algorithms, to find methods of how motion equations can be efficiently implemented even in a distributed manner. In the second approach, the aim was to generate motion equations, however more focused on automatical generation. This allows the robots to calculate autonomously the model according to the chosen topology. Additional elegant formulation technique, which uses the Lie Algebra, provides a better performance in comparison to standard techniques. Our final conclusion consists in the recommendation to merge both approaches into a single method, argued by the limited resources of multi-robot organisms and their distribution among the whole structure.

Chapter 5
Learning, Artificial Evolution and Cultural Aspects of Symbiotic Robotics

5.1 Machine Learning for Autonomous Robotics

Marc Schoenauer, Michele Sebag

In the early days of Robotics, Machine Learning (ML) used to be mentioned as a key perspective for further study; on the one hand, intelligent robots should indeed be endowed with learning abilities; on the other hand, whenever possible, manually designing competent robotic controllers is more effective than learning them (Brooks, 1986).

As fully demonstrated since the 2005 Grand Darpa Challenge (Montemerlo *et al.*, 2006) however, ML can play a central role in Autonomous Robotics. The grand Darpa Challenge 2005 was concerned with Autonomous Vehicle Driving in desert areas; the following Challenge was concerned with Urban Driving (Darpa-Urban, 2007). Another famed Robotic Challenge is the Robot Soccer Cup (Iocchi *et al.*, 2009). While robot soccer might appear to be more indirectly relevant to the everyday life than autonomous vehicle driving, the soccer game constitutes an ideal benchmark for collective decision in an uncertain and adversarial world. In all above cases, Machine Learning was the core enabling technology used to achieve individual decision making based on complex sensor data and aimed at a high level goal.

More specifically, in the last decade ML has played three main roles in Robotics, respectively related to design, dialogue, and debug.

Design. In some cases, the robot task can hardly be specified in a complete way (e.g. grasping objects of any shape and putting them in the dish washing machine (Saxena *et al.*, 2008a) or guiding visitors in a museum (Kobayashi *et al.*, 2008)). The ML-based alternative to manual design shifts the focus from designing to *training* the controller. Specifically, the robot controller is optimized to solve a restricted set of "case studies" during the training phase, and some guarantees about the generality of the acquired skills (the probability that the robot would solve other, similar, case studies) are provided.

Dialogue. ML is viewed as a core enabling technology for establishing a bridge between human and robotic worlds (Lang *et al.*, 2009). Human beings have forged

P. Levi and S. Kernbach (Eds.): Symbiotic Multi-Robot Organisms, COSMOS 7, pp. 337–433.
springerlink.com

semantic representations, e.g. a language, for their sensory patterns (Lakoff, 1987); robots need to learn how to associate these words to their own sensory patterns in order to decipher user's orders or indications.

Debug. Last, but not least, ML can be used to support the verification and debug of a robotic controller (Fox *et al.*, 2006): a simplified model of a complex controller is more amenable to *introspection*, in order to identify the causes of potential failures and/or assess an empirical validation setting.

The recent advances in, and need for, ML-based robotics have motivated a number of workshops and summer schools in the last years: IEEE-RAS / IFRR School of Robotics Science on Learning 2007, Robotics Challenges for Machine Learning at NIPS 2007, Workshop on Robotics Challenges for Machine Learning at IROS 2008, Workshop on Interactive Robot Learning 2008, From motor to interaction learning in robots workshop at IROS 2008 Robot Learning Summer School 2009, Workshop on Approaches to Sensorimotor Learning on Humanoid Robots at ICRA 2009, Workshop on Machine Learning and Data Mining for Robotics at ECML/PKDD to name a few.

This section will present some related work, without pretending to exhaustivity (Sect. 5.1.1), discuss some major issues for ML-based Autonomous Robotics (Sect. 5.1.2), and report on some work in progress done in the context of Swarm Robotics, specifically the SYMBRION/REPLICATOR projects, in Sect. 5.1.3.

5.1.1 Related Work

Most generally, ML proceeds by solving optimization problems. The state of the art in ML-based robotics will thus be naturally structured along three interdependent dimensions: the objective function (the targeted applications), the search space (the controller representation), and the algorithmic approaches (how to navigate the search space and find a quasi-optimal solution).

Algorithmically, the ML art is to define the learning criterion, i.e. the function to be optimized, depending on i) the empirical evidence, ii) the search space, iii) the computational budget and iv) the guarantees required about the optimality and generality of the solution.

5.1.1.1 Learning Robot Skills

Robotic applications are hierarchically organized, depending on the skills required to achieve the goal at hand. The primary level aims at perceptual skills; it is concerned with the structuration and interpretation of the raw sensor data. Interestingly, perceptual skills can hardly be assessed *per se*: they are essentially meant to support other skills[1].

[1] Arguably, a perceptual subsystem can be assessed in isolation, e.g. determining whether targeted objects are recognized from video data. In a robotic perspective however, perceptual skills must be assessed along an integrated setting, allowing the robot to control its moves in order to confirm/infirm its current interpretations of the sensor data (using e.g. active vision).

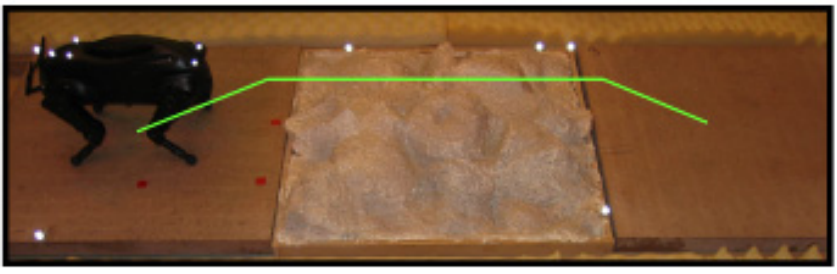


Fig. 5.1 Quadrupedal Locomotion (from (Abbeel, 2008)).

On the top of perceptual skills, are motor skills: locomotion, specifically biped or quadrupedal locomotion (see e.g., (Raibert *et al.*, 2008), Fig. 5.1) and grasping (e.g., (Saxena *et al.*, 2008b), Fig. 5.2). It must be emphasized that the difficulty of any locomotion or grasping-based task widely varies depending on the context: climbing a frozen hill requires significantly more efforts than moving on a plane ground; likewise, being able to grasp any type of cup in any position, or to grasp a moving ball, resorts to solving different sub-problems.

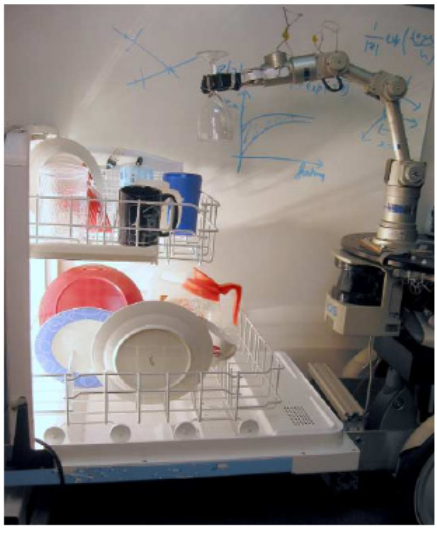


Fig. 5.2 Grasping objects (from (Saxena *et al.*, 2008b)).

One touchstone for the difficulty of the task is whether it can be solved using *reactive* strategies, or whether some type of anticipatory/-planning behaviour is required. Reactive approaches can be illustrated by the early ALVINN system (Pomerleau, 1989), pioneering the Learning by Imitation approach (Sect. 5.1.1.4). Exploiting empirical evidence gathered from the teacher's demonstration, ALVINN learned to predict the steer angle from the current road image in order to avoid obstacles; the approach was shown to be able to drive a vehicle. Based on the same principles, Le Cun showed in 2006 that convolutional neural networks can be trained from ample empirical evidence in order to achieve fast driving in non-trivial environments (LeCun *et al.*, 2005).

The limitations of reactive behaviours can be illustrated as follows. When the environment plus the robot sensors are subjected to the so-called perceptual aliasing phenomenon[2], predicting the best direction depending on the current information

[2] The robot cannot make differences between different locations on the basis of its only sensor values.

does not enable the robot to arrive at the prescribed location. Perceptual aliasing can be addressed using elaborate feature engineering, e.g. providing the robot with some information about the long-term effects of the current decisions (e.g., do not engage in a cul-de-sac). Feature engineering indeed constitutes a considerable part of the Robotics literature; how to get beyond guess-and-check approaches is a main challenge for ML-based robotics.

On the top of motor skills are more elaborate goals, involving planning at least to some extent. On the locomotion side are all activities related to navigation. While obstacle avoidance is a reactive behaviour, goals such as mowing the grass or vacuum cleaning can either be tackled using reactive stochastic strategies (Tribelhorn & Dodds, 2007) or more elaborate ones, e.g. based on building a map of the environment (Durrant-Whyte & Bailey, 2006). Along the same line are patrolling (same as mowing or cleaning, but addressed by a team of robots as opposed to, by an individual robot) and rescue (patrolling in order to find targeted patterns, e.g. individuals buried by a earthquake); and, of course, military activities. The underlying challenge in such activities is to elaborate an operational model of the visited world as complete as possible, listing all objects in the environment in a principled way, and supporting the object retrieval when needed.

On the grasping side are all activities related to assembling things, e.g. Lego building blocks in order to build a tower, a plane, and so forth. A main challenge here is to acquire the specifications of the task, using Learning by Imitation (Sect. 5.1.1.4) or by direct communication with human teachers, see below.

Besides motor skills are communication skills. As claimed by the Nobel prize winning Herbert Simon, founder of Artificial Intelligence and of the Bounded Rationality principles, *intelligence cannot be separated from social interactions*. In assembling activities for instance, the specifications of the task (*put the small blue piece in the round one*) can be acquired using multiple cues integrating visual and speech information (Aboutalib & Veloso, 2007). The point here is to ground the teacher's words (nouns and verbs) within the robot perceptual and operational apparatus.

On the top of communication skills are collective tasks, e.g. soccer playing. After H. Simon again, robot soccer demonstrates a "social grammar"; mastering this grammar definitely is one of the biggest Robotic challenges.

5.1.1.2 Controller Representation

The controller space, also referred to as search space, is a space of functions mapping the raw sensor data and possibly the internal state values of the robot, onto actuator values. This mapping can be deterministic (e.g. neural net) or probabilistic (e.g. Markov Decision Process).

Most commonly, the raw sensor data is pre-processed. The pre-processing step can correspond to (manual) feature engineering, defining intuitive state variables in order to facilitate the controller design (Bain & Sammut, 1995). Another motivation for sensor pre-processing is dimensionality reduction. The sensor data (data stream in $\mathbf{R}^D$) usually lives in a much smaller space dimension ($\mathbf{R}^d, d << D$) since

sensor values (e.g. camera pixels) are correlated in the sense that they provide different information about the same robot environment. Getting rid of the sensor redundancy is beneficial in two ways; on the one hand, dimensionality reduction improves the quality of the information provided to the controller, through removing the sensor noise. On the other hand, the amount of empirical evidence needed to train a controller up to a certain degree of accuracy exponentially increases with the dimension of the input space, everything else being equal. In a pre-processing stage, the raw sensor data thus is decorrelated and denoised, using either Principal Component or Independent Component Analysis (Hyvarinen *et al.*, 2001), as in (Calinon & Billard, 2005), or non-linear dimensionality reduction techniques, e.g. Isomap (Tenenbaum *et al.*, 2000). Note that sensor pre-processing, also referred to as sensor fusion (see Sect. 3.2), is done on-line. It might be required to replace well-founded dimensionality reduction algorithms with approximations thereof.

Various controller spaces have been considered in the literature, ranging from rule-sets to self-organized maps, (recurrent) neural nets, graphical models, Hidden Markov Models (HMM) and Partially Observable Markov Decision Processes (POMDP). The choice of the search space depends on the targeted application, on the desired properties of the solution (e.g. rule-sets are more easily interpreted than neural nets) and specifically on the prior knowledge of the designer. For instance, the invariance properties of perceptual skills (the interpretation of an image does not change much if the image is slightly rotated, translated, or undergoes a homothety) can be directly encoded in the topology of neural nets, as shown by Le Cun's convolutional nets (LeCun *et al.*, 2005). The spatial context of the robot can be used to guide the structure of a graphical model. The robot dynamics can be encoded through the HMM (Gienger *et al.*, 2008). Rule-sets enable to directly exploit fragments of expert knowledge (Kavka *et al.*, 2005); they also are easier to debug than e.g., neural nets. Oscillators have been used for locomotion. Self-organized maps arguably enable non-linear input-output mappings while supporting dimensionality reduction (Barreto *et al.*, 2003).

A very general framework is that of Markov Decision Processes (MDP). In MDPs, the world model is made of a set of states $\mathscr{S}$, a set of actions $\mathscr{A}$ connecting the states, and a transition function indicating the probability of arriving at state s' by selecting action a when in state s. The task at hand is further modelled by a reward function, associating a quality ($\in \mathbf{R}$) to each state. Defining a policy as a function mapping each state onto an action, the MDP problem amounts to finding an optimal policy, that is, one maximizing the value of the reward expectation on a finite, or an infinite, time horizon[3].

How to build such an optimal policy, aka controller, is tackled by Reinforcement Learning (Sect. 5.1.1.3). Another main approach, referred to as Learning by Imitation (Sect. 5.1.1.4), exploits the teacher's demonstrations. Finally, Inverse Optimal Control bridges the gap between RL and LI (Sect. 5.1.1.5).

[3] In the infinite horizon case, one considers discounted rewards to enforce a finite expectation.

5.1.1.3 Reinforcement Learning

Reinforcement Learning (RL) (Sutton & Barto, 1998; Ng & Russell, 2000) aims at optimizing the reward expectation along the considered time horizon: by construction, the best policy is the one maximizing the reward expectation. The RL principle is based on iteratively computing some value function, associating to each state s (each pair state, action s,a) the reward expectation reaped when the robot initial state is s (resp. when the robot initial state and action are s and a).

The main strength of RL is to provide a sound framework for building provably optimal policies. In counterpart, this nice setting is expensive in size and/or efforts; some discretization is needed to map a continuous world onto a discrete set of states; the number of states exponentially increases with the number of relevant features. The main research priority in RL thus is: scalability.

A key issue, referred to as Function Approximator, uses various supervised ML settings to estimate the value function $Q(s,a)$ from the empirical evidence, using look-up tables, nearest neighbours, neural networks, decision trees, or tile coding (CMAC, or Cerebellar model arithmetic computer).

Another issue is to face the curse of dimensionality, through reducing the state space and/or the action space. For instance, the state space can be reduced through abstraction. The simplest type of abstraction is to discretise a continuous search space. When the state space is described through the (continuous) sensor value vector, the discretization aims at minimising the variance of the value function (Munos & Moore, 2002). Another possibility is to generalise across states.

Independently, the action space can also be reduced. One possibility is to consider "options" (Sutton & Barto, 1998; Precup *et al.*, 2006). An option is a macro-action, or sequence of actions; an option o is a triple (A_o, π_o, β_o) where A_o is the set of states where option o can be selected, π_o is the policy followed during option o, and β_o is the probability of terminating option o in each state ($\beta_o : \mathscr{S} \mapsto [0,1]$). Formally, the introduction of options leads to considering Semi-Markov Decision Processes. Practically, options correspond to sub-goals; their introduction leads to implementing modular controllers. The benefits of options are those of modularity: reusability of partial controllers; faster learning; better understandability/inspection of the solution policy.

Predictive State Representations (PSR) (Littman *et al.*, 2002) offer a new framework for representing states and actions. The challenge in some sense is to overcome the long known failure of AI to provide sound and robust descriptions of useful concepts. For instance, providing a comprehensive description of what a "chair" is was found an elusive task, due to the number of possible exceptions, the diversity of the contexts, and more generally, the openness of the world. A way out of this trap is offered by the so-called sensorimotor contingencies, inspired from Cognitive Sciences: a chair is characterised by the fact that, when I sit upon it, I don't fall down. More generally, a concept is viewed as an elementary predictive model: if action a is applied, and the result is such and such, then the robot was in state s. Formally, PSRs are characterised in terms of core tests (discovered from traces); learning the world model corresponds to associating to each history a probability

of satisfying every core test. In other words, a state is described as the conditional probability of the core tests. In a theoretical perspective, PSRs with k core tests enable to represent dynamical systems of dimension k (strictly including POMDP with k states). In practice, two main issues are open: i) how to define the core tests; ii) shouldn't one go beyond linear PSRs (e.g. using Predictive Linear Gaussian instead (Wingate & Singh, 2006)).

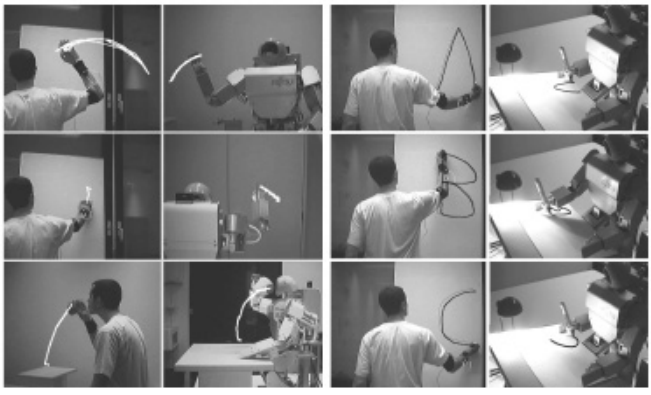
5.1.1.4 Learning by Imitation

While RL provides sound guarantees of optimality through a sufficient exploration of the search space, Learning by Imitation (LbI) (Pomerleau, 1989; Bain & Sammut, 1995; LeCun *et al.*, 2005) is provided with an example of allegedly optimal behaviour, namely the teacher's traces (e.g. acquired by motion capture). These traces define a supervised learning problem: learn to associate the desired action to each sensor input. For instance (Bain & Sammut, 1995) aim at "cloning" a plane pilot; (LeCun *et al.*, 2005) exploit driving traces, where each video frame is associated to the steering angle of the driver; the controller learned is demonstrated on a four wheeled robot, driving in the forest. As mentioned earlier on however, this supervised learning setting is limited to training reactive controllers: when considering a deliberative task, a low average prediction error does not guarantee that the final performance will be satisfactory (missing one single critical step is enough to fail the job). Another limitation of LbI is that the teacher's traces reflect both the teacher's errors and her efforts to correct these errors; the reproduction of the traces *verbatim* thus becomes a deceptive goal from the perspective of effective control. Along the same lines, the variability of the traces (e.g. different teachers might operate with different tempos) requires some generalization to be done. For instance, the demonstrations provided by different teachers, e.g. knocking at the door (Fig. 5.3) are generalized using an HMM framework.

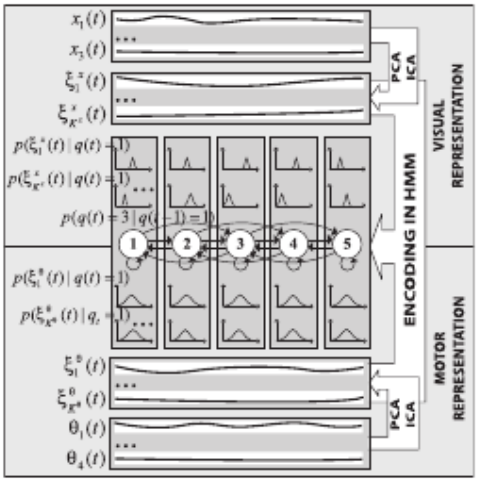
Yet another difficulty is that human beings and robots operate in different perceptual and operational spaces. Typically, human and robot arms might differ by their degrees of freedom (Calinon & Billard, 2005). Human teachers must be provided with more precise visual cues than will be given to the robot on-line (e.g. using camera with better definition) in order to enable demonstration.

One of the most effective heuristics involved in the Darpa Challenge winning Stanley vehicle (Montemerlo *et al.*, 2006) participates to some extent to Learning by Imitation. Stanley, aimed at autonomous fast driving in the desert, is endowed with a variety of sensors, including high-definition cameras and long-distance infra-red sensors. Clearly, cameras provide a much richer information than infra-red sensors; in the meanwhile, they are limited to near-environment.

One of the main sub-problems faced by STANLEY is to tell drivable from non-drivable (bush, rocks, water) ground, in order to reach the target location as fast as possible. In the preliminary training phase, much data has been gathered to provide a "sufficiently complete" catalogue of the diverse situations the robot vehicle can meet, and train efficient classifiers. As could have been expected however, there is no such thing as a complete catalogue; lifelong learning thus is required.



(a) Demonstration



(b) Input data

Fig. 5.3 Learning by Imitation (from (Calinon & Billard, 2005)).

A partial solution can be found, based on high-definition cameras; these provide enough information to reliably classify the ground in front of the vehicle. The limitation is that the prediction, being available only when the vehicle is close to the zone, does not enable fast driving.

This limitation is addressed as follows. At any time t, the robot is provided detailed information about its close environment, and coarse information about farther away locations x. The close environment can thus be reliably classified as, say, road and non-road. After a while, some formerly far away locations x have become close ones. For some locations, the robot is thus provided with: i) the coarse description (available at t); ii) the high-definition description (available at $t + \Delta$); ; iii) and thus the label, road or non-road, which can be inferred from the high-definition location.

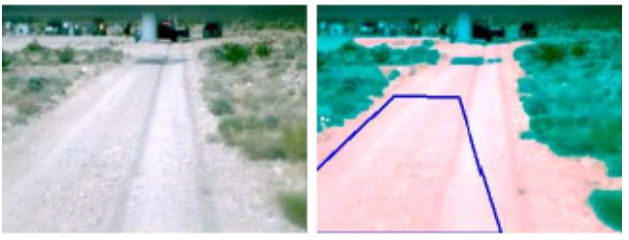


Fig. 5.4 Grand Darpa Challenge: The environment (from (Montemerlo *et al.*, 2006)).

A problem analogous to Learning by Imitation can thus be formulated, where the input is made of the coarse description of location x, and the associated label is given by the "expert" classifier, operating on the detailed description of x. An embedded learning component seamlessly resolves the long-range classification problem, estimating whether a far-away location is road or non-road. While the obtained classifier is only available at $t + \Delta + \textit{Learning time}$, after the landscape continuity assumption this classifier is competent to classify the currently far-away locations.

The bottom-line is that, although long-range classifiers are less reliable than the short-range ones, they nevertheless allow the vehicle for speeding ahead toward the safe locations. More precisely, the detection range goes to about 70 meters, as opposed to 20 meters when using the high definition camera only (Thrun *et al.*, 2005a).

5.1.1.5 Inverse Optimal Control

Inverse Reinforcement Learning (IRL), also referred to as Inverse Optimal Control, bridges the gap between Reinforcement Learning and Learning by Imitation by using the teacher's demonstrations to infer some reward function; this reward function makes it possible to use the RL machinery in order to mimic and/or generalize the teacher's behaviour.

IRL relies on two main assumptions. Firstly, it assumes that the world space (sensor $\times$ action space) can be mapped one-to-one onto some feature space. In a parking task for instance, the features include the curvature, smoothness, distance to obstacles, and alignment with principal directions. Like early robotic approaches (Brooks, 1986), the approach thus critically depends on feature engineering.

Secondly, it assumes that the targeted behaviour is obtained by minimizing some cost function over the trajectory in the feature space. The linearity of the cost function was assumed in early work devoted to IRL (Abbeel, 2008); the linearity assumption has been relaxed later on (Ratliff *et al.*, 2009).

The IRL goal thus is to learn the reward function, based on the assumption that the teacher's demonstration reflects an optimal policy, that is, it must get a greater cumulative reward than all other controllers in the search space. For the sake of scalability (as the size of the controller space is exponential in the size of the state and action space), the constraint is equivalently translated into: the teacher's

demonstration should get a greater cumulative reward than the *best* controller according to the current reward function.

IRL thus proceeds iteratively. Let ξ denote the sequence of (state, action) corresponding to the teacher's trace. Given an initial reward function r_0, let π_0 denote the optimal policy based on r_0. At every step t, let $R_t(\pi_t)$ define the cumulative reward associated to the (state, action) sequence obtained by following π_t from the initial state, after r_t; let $R_t(\xi)$ be likewise the cumulative reward associated to the (state,action) sequence coding the teacher's demonstration. If $R_t(\pi_t)$ is close to $R_t(\xi)$, the process stops; otherwise, the reward function is modified in order to account for the fact that π_t is not an optimal policy. Formally let S denote the feature space, and let $\xi \in \{0,1\}^S$ be the indicator vector of the teacher trace ($\xi_j = 1$ iff s_j belongs to the trace). In a linear setting, the reward function is given as a weight vector w_t, with

$$R_t(\xi) = < w, \xi >$$

The optimisation problem can be formalised as, where π_t denotes the thus-far found policies:

$$\text{Find } w_t \text{ maximising } \gamma \, s.t. \, ||w|| = 1, R_t(\xi) \geq R_t(\pi'_t) + \gamma, \text{ for } 0 \leq t' \leq t-1$$

A new policy π_t is learned after reward w_t, and the process is iterated until $R_t(\pi_t)$ is sufficiently close to $R_t(\xi)$. Contrasting with the myopic decision problem addressed by Learning by Imitation (find the appropriate action depending on the current state), IRL thus considers and optimizes the whole (state,action) sequence.

The formalization can be further improved within the Maximum Margin Planning framework (Ratliff *et al.*, 2009), associating to each policy/trajectory π and each demonstration $\xi_i, i = 1 \ldots K$ a loss function $\ell_i(\pi)$. Let π_i^* denote the optimal policy in the sense of the reward minus loss function:

$$\pi_i^* = min_\pi \; < w, \pi > -\ell_i(\pi)$$

Then the margin-based formulation of the reward function is such that

$$< w, \pi > \; \leq \; < w, \pi_i^* > -\ell_i(\pi_i^*), \forall i = 1 \ldots K$$

Avoiding trivial solutions related to small weight vectors w, and associating slack variable ζ_i to each constraint above, it comes:

$$\begin{array}{ll} \text{Find} & \min \frac{1}{2}||w||^2 + \frac{1}{K}\sum_i \zeta_i \\ \text{subject to} & < w, \pi > \; \leq \; < w, \pi_i^* > -\ell_i(\pi_i^*) + \zeta_i, \forall i = 1 \ldots K \end{array}$$

In brief, the Inverse Reinforcement Learning claim is that one should explore the reward function space, rather than the policy space. The main rationale for this claim is that, if one gets the reward function right, then RL provides guarantees about the optimality of the policy; further, as shown by Abbeel (Abbeel, 2008), the number of steps and examples in the iterative IRL process can be upper bounded.

5.1.2 Challenges for ML-Based Robotics

The quality of the learning output primarily depends on the quality of its input. The quality of the information provided to the robot, specifically whether the world is observable or partially observable, on the one hand, and whether it is deterministic, on the other hand (Taylor *et al.*, 2006) must govern the complexity of the controller model and the experimental setting of the training process. An additional difficulty is related to the use of simulators, providing a fast noisy approximation of both the environment and the robot behaviours. The first challenge for ML-based Robotics thus is to devise an integrated training process, combining *in-silico* and *in-situ* training.

Another key issue is to assess the robot behaviour. As amply demonstrated in Evolutionary Robotics (Nolfi & Floreano, 2000b), the assessment criterion must enable to gradually improving the robot behaviour; Needle in the Haystack-like problems should be avoided. How to guide the learning search through the interaction with the designers, thus constitutes a main methodological challenge for ML-based Robotics.

5.1.2.1 Quality of the Controller Input

Autonomous controller design mainly comes into two flavours, *in-situ* and *in-silico* training. *In-silico* approaches extensively rely on robotic simulators, encoding a world model and specifically sensor and actuator models; *in-silico* trained controllers reflect the quality of the underlying simulators. Acquiring reliable models of the sensor and actuator devices however raises significant difficulties, e.g. relatively to the noise model[4] and the many parameters involved in real world modelling (e.g., light conditions and texture of the obstacles for visual sensors). Physics-compliant simulators (e.g. modeling elastic shocks, gravity, etc) are slow, to such an extent that simulations might take about as long as *in-situ* experiments.

In short, *in-silico* approaches offer either cheap and unreliable, or computationally expensive and more accurate, environments for robotic controller training. In both cases, controllers are prone to suffer from the so-called *Reality Gap*: their actual performance does not match the simulated one, in other words they overfit the simulators[5].

In opposition, *in-situ* approaches always are time-demanding: experiments take time and they require the full participation of the human designer (Floreano & Mondada, 1994; Nolfi & Floreano, 2000a). Furthermore, they put the robotic devices at high stress, increasing the chances of failure. The exploration of the environment and the controller space also requires special care to preserve the

[4] It is widely acknowledged that sensors provide noisy information; actuators likewise are noisy. The noise model however proves to be elusive (Thrun *et al.*, 2005a).

[5] A possibility investigated by (Kolter *et al.*, 2008) is based on using an ensemble of simulators; the ensemble of controllers learned from these simulators is used to reduce the dimensionality of the controller search space.

robot integrity. Lastly, *in-situ* training itself does not necessarily offer decent guarantees about the controller generality, for two reasons. On the one hand, robot sensors and actuators might sensibly differ from one robot to another; the controller trained on one physical robot might present some inaccuracies when transfered on another robot. On the other hand, autonomous robots cannot be trained under all possible experimental conditions; a modest change in the light conditions might hinder the effectiveness of the robot controller.

One promising direction for handling the *in-silico* vs *in-situ* (ISIS) dilemma is to consider that the Reality Gap (RG) is unavoidable, whatever the training mode of the controller is. Accordingly, a core task in Autonomic Robotics becomes to deal with the RG phenomenon. Interestingly, Autonomic Computing (Kephart & Chess, 2003) faces similar challenges, related to building self-adaptive and self-healing systems. A key milestone toward such autonomic systems is to make them self-aware, i.e. able to anticipate the effects of their decisions. Some promising steps toward self-awareness, specifically self-modeling resilient robots have been done by (Bongard *et al.*, 2006). This approach can be extended along the lines of multi-task aka transfer learning (Bickel *et al.*, 2009), offering a principled learning approach when the training environment might be different from the targeted environment.

5.1.2.2 Quality of a Controller

A second core issue is to provide the robot with some criterion of quality, decently encoding the desired behaviour. How critical the design of a robotic fitness function is (all the more so due to Evolutionary Opportunism), has been extensively discussed in Evolutionary Robotics (Nolfi & Floreano, 2000a); the design of reward functions in Reinforcement Learning raises similar difficulties. Specifically, the fitness/reward function defines an optimization problem, with two requirements. On the one hand, the optimization problem should be tractable (avoiding Needle-In-The-Haystack-like optimization landscapes). In practice, the criterion should enable making differences between inappropriate robotic behaviours and *very* inappropriate behaviours, conducive to the discovery of appropriate ones. On the other hand, the criterion should not admit trivial and undesirable solutions; typically, an undesirable way of achieving obstacle avoidance is to stay motionless, or to rotate without advancing (Nolfi & Floreano, 2000a).

As shown in Sect. 5.1.1.4, Learning by Imitation (LbI) is a promising alternative to the definition of a quality criterion, in the spirit of Machine Learning: instead of providing rules or criteria, the human teacher provides examples. The literature (Bain & Sammut, 1995; Abbeel & Ng, 2004; Calinon & Billard, 2005) suggests that LbI both addresses some known difficulties and raises some new difficulties. One of those is the presence of noise in the teacher traces, making it necessary to edit the teacher traces in order to effectively guide the robot. While some LbI limitations are addressed by Inverse Reinforcement Learning (IRL; Sect. 5.1.1.5), the latter approach still depends on the teacher's expertise. A Game Theoretical approach developed by (Syed & Schapire, 2008; Syed *et al.*, 2008) has been shown to

overcome the teacher's weaknesses, subject to the identification of the desired features (e.g. speed, obstacle avoidance,..); basically, the ultimate policy maximizes the minimal reward expectation over all reward functions described as weighted combinations of the features.

Another approach directly aims at learning the expert's preferences, through either interactive optimization (Llorà *et al.*, 2005), or preference learning (Cohen, 2000; Freund *et al.*, 2003; Burges *et al.*, 2007). Basically, the idea is to present the designer with several behaviours, ask for a partial ordering of these behaviours and infer a preference model from this partial order. Such a preference model corresponds to the sought reward or fitness function. Interestingly, preference learning could be further coupled with active learning (Dasgupta, 2006), allowing the learning controller to ask the teacher about the most appropriate actions in critical situations.

5.1.3 The WOALA Scheme

This section presents a Machine Learning-based scheme for Swarm Robotics, devised in the SYMBRION/REPLICATOR framework, and incorporating some of the ideas described above. The specific principles and requirements guiding the presented scheme, called WOALA, are first discussed. An overview of WOALA is presented together with a proof of concept of the approach. aimed at making the distinction between seeing an obstacle and seeing another robot. It is worth emphasizing that making this distinction is not a trivial problem[6], while it is a critical milestone in a swarm robotic framework.

In the following, *perceptual apparatus* is meant as a set of descriptive features, hierarchically built on the raw sensor data; *operational apparatus* is a set of actuator patterns, related to the perceptual apparatus.

5.1.3.1 Working Hypotheses

The WOALA scheme has been devised with two main issues in mind. The first one regards the division of labour between programming and learning, or the distinction between "innate" vs "acquired" skills. The second one aims at the scalability of the approach w.r.t. the complexity of the targeted behaviour, through capitalizing the robotic know-how (perceptual, motor, and/or deliberative skills) acquired along the process.

One of the central issues in Autonomous Robotics is to distinguish between what should be given to the robot ("innate" skills) and what should be learned ("acquired" skills). Clearly, there is no point in learning skills which can be easily and efficiently programmed. While the distinction between innate and acquired skills has long been debated and various interpretations have been proposed in Ethology (Lorenz, 1941),

[6] Making the distinction between mobile and motionless obstacles in all generality requires active vision, that is, planning skills.

their relevance to Robotics remains unclear. Biological entities participate in an integrated, partially observable, spatio-temporal system; the goal of Ethology is to analyse how this system has been deployed "from scratch". Quite the contrary, the goal of Robotics is to create (mechatronical) entities, with a focused goal and controlled initial conditions.

Nevertheless, several milestones for Autonomous Robotics have been defined from Ethology studies, ranging from foraging skills (Koza, 1992) to latent learning (Tolman, 1948; Lanzi, 2000; Hartland *et al.*, 2009). The proposed approach is inspired by a particular experiment reported by Konrad Lorenz, shedding some light on the distinction between innate and acquired skills, relevant to our purpose. After (Lorenz, 1941), the young goose is programmed to follow its mother (innate behaviour); the "mother" pattern however is acquired, to such an extent that the young goose might mistake the ethologist for its mother and follow him, under appropriate experimental conditions. In these experiments, it is suggested that the operational apparatus is built-in (innate following behaviour) whereas the perceptual apparatus (the mother pattern) is learned along a well-defined scenario (a shape present at the young goose' birth, moving with an appropriate speed).

Another major issue regards the computational effort, and even more importantly, the human design effort, required to acquire or specify the diverse skills (perceptual, motor, deliberative) involved in Autonomous Robotics. The early robotic trend, e.g. illustrated by Brook (Brooks, 1986), used to define specific acquisition sub-problems, the solutions of which were integrated within a global subsumption architecture. The ML-based approach defines generic tools addressing diverse goals; for instance the optimal inverse control approach proposed by Ratliff et al (Ratliff *et al.*, 2009) can indifferently be applied to quadrupedal locomotion, car parking or object grasping problems. None of these approaches however seems to be able to gracefully scale up, due to intrinsic limitations. On the subsumption side, skills are independently acquired, lending themselves to a capitalization of the know-how; the integration of the know-how however relies on the subsumption architecture, which is manually engineered. On the ML-side, the stress is put on the genericity of the algorithms; the approach however relies on the choice of an appropriate representation (state and action spaces), which governs the efficiency of the approach. Further, how to capitalize the know-how – how to reuse former controllers – is not considered.

A more promising framework, scalability-wise, is offered by the Action Selection framework (Humphrys, 1997; Godzik *et al.*, 2003). This framework involves a generic action space; typically, elementary actions correspond to the basic actuator patterns (go ahead, left, right; grasp). On the top of this initial action space, the AS framework enables to define simple controllers (obstacle avoidance, light search). The AS framework however lends itself to the capitalization of the know-how, in the sense that formerly acquired controllers can be considered as (macro-)actions and selected when tackling new goals. The AS framework thus enables one to deal

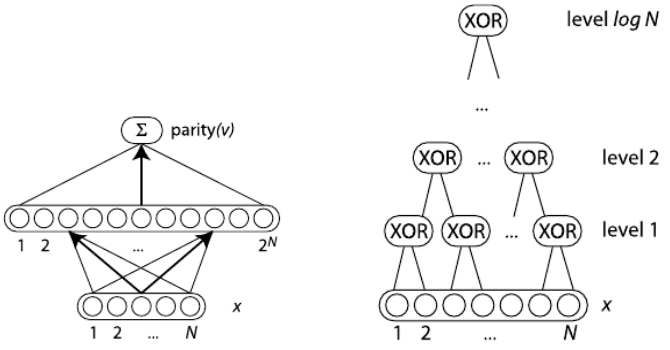


Fig. 5.5 Shallow vs Deep Structures (from (Larochelle *et al.*, 2007)).

with a hierarchically structured action space, where i-level actions are defined using $i-1$-level actions as building blocks[7].

The fact that functional spaces with deep structures are (exponentially) more compact than shallow ones everything else being equal has been known for long (Hastad, n.d.), and can be illustrated on the toy n-bit parity problem (the target function value is 1 iff the sum of the n bits is even). In Disjunctive Normal Form (the disjunction of conjunctive monoms, corresponding to a shallow network), n-bit parity is expressed through 2^{n-1} monoms (hidden neurons). The use of nested expressions (a network with $log\, n$ layers) makes it feasible to characterize the n-bit parity with n neurons (Fig. 5.5).

As argued by Y. Bengio and G. Hinton (Larochelle *et al.*, 2007; Hinton *et al.*, 2006), the major reason why shallow topologies/representations have been preferred over deep ones for over two decades despite their comparatively poor expressiveness is their better controllability. More precisely, supervised learning within a deep network architecture defines an optimization problem plagued with local optima, many of which have a quite poor generalization performance. This limitation has been sidestepped in the Deep Network framework (Hinton *et al.*, 2006; Larochelle *et al.*, 2007), iteratively building the hidden layers of the Deep Networks using an auxiliary optimization criterion[8].

[7] It might be objected that complex tasks are not optimally addressed by concatenating simple actions, in the general case. The considered hierarchical structure however makes it feasible to find a satisfying solution for a complex problem; an eventual fine-tuning phase will enable to adjust and polish the simple actions involved with the complex goal in mind. In brief, modularity is viewed as a core enabling principle to achieve complex goals (Dawkins, 1986 - 1996).

[8] Formally, the i-th layer of the network is defined as an (invertible) encoding of the $i-1$-layer, the 0-th layer corresponding to the initial representation of the data. The actual supervised learning task is taken into account when the current layer of the Deep Network is deemed to provide an adequate representation of the available information. The interested reader is referred to (Hinton *et al.*, 2006; Larochelle *et al.*, 2007) for a comprehensive presentation of Deep Networks.

Along this line, it is suggested that complex goals can be tackled by i) using a single generic learning algorithm; ii) iteratively building more complex representations (the successive topology layers); iii) solving a sequence of learning goals (the *i*-th layer encodes the $i-1$-th layer). Such a strategy is remotely reminiscent of Piaget's theories about cognitive development (Piaget, 1937), where concepts are iteratively acquired, and each set of concepts (e.g. space or time invariance) paves the way for representing and acquiring more complex concepts.

5.1.3.2 Overview of WOALA

Taking inspiration from the above mentioned theories and approaches, the proposed scheme aims at gradually acquiring a complex robotic perceptual and operational apparatus, by applying generic ML algorithms to robotic logs.

Formally, the division of labour between "innate" and "acquired" development goes as follows. On the innate side, the designer is in charge of devising a scenario conducive to the observation of "interesting" events and sequences of events in the scope of available (programmed or evolved) controllers. Running the controller after this scenario generates a log file, recording the sensor and actuator data experienced by the robot. The designer additionally annotates the log, labelling some (sequence of) time steps as corresponding to interesting events.

On the acquired side, the annotated log is processed, using available features to describe the learning instances; these instances, together with the labels provided by the oracle, define learning problems. The resolution of these learning problems derives hypotheses, enriching the perceptual and operational apparatus of the robot. The increase in complexity is achieved by i) using the perceptual features acquired in the previous phase to represent the learning examples; ii) learning gradually more complex target concepts, using the human designer as oracle.

Specifically, the WOALA scheme is a 5-step process: Write-Observe-Annotate-Learn-Assess.

Write — In the first step, the human designer writes a controller π based on the existing perceptual and operational apparatus. Additionally, the designer defines a scenario conducive to "interesting events" (see below). Initially, the controller might be a standard Braitenberg controller (Braitenberg, 1984), simply achieving obstacle avoidance. The proposed scenarios are detailed in Sects. 5.1.4.3 and 5.1.4.4.

Observe — In the second step, controller π is launched along the defined scenario, either in simulation or in situ. The robotic log is recorded, indicating the perceptual and operational events holding true for each time step, together with the events prescribed from the scenario.

Initially, the robotic log only stores the raw sensor and actuator data stream perceived and acted by the robot.

Annotate — In the third step, the robotic log is annotated after the prior knowledge, either from the available resources (Sect. 5.1.4.3, Sect. 5.1.4.4) or from the designer. In the first case, the annotations define a standard

supervised learning problem: this (sequence of) sensorimotor data corresponds to seeing i/ a wall, ii/ another robot. In the second case, the annotations define a preference learning problem (Cohen, 2000; Freund *et al.*, 2003; Burges *et al.*, 2007): after the designer, this sequence of sensorimotor data is more appropriate (to the ultimate goal) than this other sequence.

Initially, some resources will be developed to enable the automatic annotation of simple target concepts (Sect. 5.1.4.3).

Learn In the fourth step, the learning problems set in the former step are solved and hypotheses are built. Notably, hypotheses built from a supervised learning setting can be viewed as virtual or educated sensors; they correspond to perceptual primitives (I see a wall). Hypotheses built from a learning to rank setting correspond to surrogate rewards. Such surrogate rewards contrast with the Inverse Optimal Control (Sect. 5.1.1.5) in two ways. Firstly, they are learned by comparing fragments of robotic behaviours, as opposed to, from the teacher's demonstrations; secondly, the reward applies to a sequence of time steps.

Assess In the fifth step, the hypotheses built in the previous step are assessed, typically using a test set extracted from the robotic log. The best hypotheses are added to the robot perceptual and operational apparatus, and they become available to write another controller.

The originality of the WOALA scheme lies in the division of labour between the innate and the acquired elements of the controller. The acquired (learned) elements constitute a hierarchically structured set of perceptual primitives and rewards.

The acquisition of these primitives relies on the human designer in two different ways; firstly, the designer puts the robot in situation (writing the controller and the scenario); secondly, she provides a feedback through annotating the empirical evidence. In some sense, this approach closely reflects Herbert Simon's claim (Sect. 5.1.1.1), rooting robot intelligence in social interactions with the designer.

The main limitation of the approach regards its bootstrap, i.e. the definition of a scenario and a controller conducive to the observation of interesting events, and the learning of the first perceptual features and rewards.

5.1.4 First Experiments with WOALA

In order to assess the viability of the proposed approach and the validity of the WOALA scheme, some simple experiments involving the complete WOALA loop have been designed. The aim is to learn some advanced primitives, namely distinguishing between seeing a robot and seeing a wall, which are mandatory in the context of swarm or cellular robotics.

The challenge lies in dealing with the most simple model of robot that has been proposed within the project. While this robot should be able to detect other similar robots in order to either adopt a swarm behaviour, or to join a multi-cellular organism, it is only endowed with 2 infra-red sensors, with a limited focus and range.

The methodology detailed below is to gather logs of wandering robots, to filter out all non-events (the robot sees nothing at all), to label the (sequence of) sensorimotor data on the basis of the available resources, to acquire a hypothesis about the target concepts, to test this hypothesis and use it to enrich the available resources.

5.1.4.1 Getting Started

A randomized Braitenberg controller was used for the *Write* step of all initial experiments. Whenever one sensor detects some obstacle, the robot starts rotating to avoid it[9], either leftward or rightward for symmetry reasons. The total rotation angle is uniformly drawn in $[0, 180]$. The rationale for such randomization is to prevent the robot from engaging into periodic behaviours, e.g. oscillating in a corner of the arena.

A mechanism allowing the designer to take control of the robot independently of the Braitenberg controller has also been implemented. Specifically, the designer controls the robot moves from a joystick or from the keyboard. Such a manual control is useful in some scenarios, e.g. to ensure that desirable events (two robots meeting) will occur sufficiently often; it will also be used in the preference learning step of the WOALA scheme (on-going experiments not described here).

Before describing the actual experiments, next section will briefly introduce the learning algorithm that has been used for all learning tasks involved in this work.

5.1.4.2 Learning Algorithm

While many learning algorithms (decision trees or random forests (Breiman, 2001), neural nets (LeCun *et al.*, 2005), Support Vector Machines (Cristianini & Shawe-Taylor, 2000)) can be used in principle, an any-time and frugal learning algorithm is needed to address the constraints of embedded robotics. The results presented below are based on the ROGER algorithm (Jong *et al.*, 2004), using Evolution Strategies to optimize the Area Under the ROC curve, also known as Mann Wilcoxon criterion.

Learning criterion: Formally, let us consider a training set $\mathscr{E}$

$$\mathscr{E} = \{(x_i, y_i), i = 1 \ldots n, x_i \in \mathbf{R}^d, y_i \in \{-1, 1\}\}$$

where x_i denotes the description of the i-th example (a vector of $\mathbf{R}^d$) and y_i is the associated label[10]. Let h define a hypothesis mapping $\mathbf{R}^d$ onto R. The Mann Wilcoxon

[9] An additional difficulty is raised by the particular motor system of the robot, which is moved by two screw-drive wheels (see Sect. 2.4, Fig. 2.29 right). While these wheels allow the robot to make a smooth move on its left, a complete stop and backward rotation are required to perform a right-turn. It is hence impossible to get a turning speed proportional to some sensor-based value, and the standard Braitenberg controller had to be modified accordingly.

[10] To fix the ideas, the dimension d in the reported experiments is 120, corresponding to 2 sensors and 2 actuators along 30 time steps.

criterion computes the fraction of examples x_i, x_j such that x_i is a positive example, x_j a negative one, and $h(x_i)$ is greater than $h(x_j)$:

$$MW(h) = \frac{|\{(x_i,x_j)\ s.t.\ y_i > y_j, h(x_i) > h(x_j)\}|}{|Z|}$$

where Z denotes the number of pairs of (negative, positive) examples. As shown by (Rosset, 2004), this criterion is stable; it is quadratic w.r.t. the number of examples whereas the standard misclassification cost is linear with the number of examples).

The computation complexity of $MW(h)$ is actually $\mathcal{O}(n \log n)$ (examples are ordered w.r.t. their h value to ease the computation of the criterion).

Hypothesis model: Several hypothesis spaces have been considered, including Echo State Networks (Hartland *et al.*, 2009). In the reported experiments, a hypothesis is described from two vectors w and c each in $\mathbf{R}^d$

$$h(x) = \sum_{i=1}^{d} w_i |x_i - c_i|$$

The main rationale for this choice of hypothesis model is to allow for non linear hypothesis while preserving a linear complexity w.r.t. the size of the input space (as opposed to e.g. standard neural nets).

Hypothesis learning: The maximization of the Mann Wilcoxon criterion over the hypothesis space described above amounts to standard parametric optimization. Due to the high number of local optima, it is performed using a standard self-adaptive non-isotropic Evolution Strategy (Bäck *et al.*, 1997).

Once a good (with respect to the Mann Wilcoxon criterion) hypothesis h has been found, a threshold τ is determined in order to use h for further classification purposes: an unknown example x will be classified as positive if $h(x) > \tau$, negative otherwise. The value of τ is determined so as to minimize the classification error (i.e. to maximize the sum of the number of true positive and true negative) over the training set.

The confidence of the classification is further assessed from the margin of $h(x)$: the higher $|h(x) - \tau|$, the more certain the classification is. Formally, the margin $mh(x)$ is defined as:

$$mh(x) = \frac{h(x) - \tau}{min_{i=1\ldots n} h(x_i) - \tau}$$

Ensemble learning: It is widely known in the Machine Learning community that ensembles of hypotheses can often be more robust than single accurate hypotheses – this is the ground basis for techniques such as bagging (Shapire *et al.*, 1997; Derbeko *et al.*, 2002) and boosting (Freund & Shapire, 1996; Freund *et al.*, 2003). The use of evolutionary optimization as learning engine gives a source of variability "for free" (Gagné *et al.*, 2007), which is exploited to independently extract 15 hypotheses from each training set.

The predicted class is then obtained by summing the margin $mh_i(x)$ over all generated hypotheses h_i. A confidence threshold τ_g is used to classify x as unknown if the sum of $mh_i(x)$ is less than τ_g:

$$class(x) = \begin{cases} unknown & \text{if } |\sum_i mh_i(x)| < \tau_g \\ 1 & \text{if } \sum_i mh_i(x) > \tau_g \\ -1 & \text{if } \sum_i mh_i(x) < -\tau_g \end{cases}$$

5.1.4.3 Building Resources

The annotation step (Sect. 5.1.3.2) aims at labelling the log events according to the targeted concept. Whereas manually annotating the events is tedious whatever the scenario, it is hardly feasible when the annotation is related to the perception of the robot (as opposed to its visible behaviour). Indeed, a human being can hardly know *what it is like to be a specific robot* (Nagel, 1974); one can hardly guess whether the robot sees the wall or another robot in any given situation.

As a starting point, the only reliable primitive is "I see something", that holds true whenever the value of the infrared sensors is different enough from the default value.

In order however to reach the target goal, being able to tell a wall from another robot, a preliminary phase is performed in order to acquire resources ("pseudo-ground truth" or oracles), which will be used to automatically annotate the robotic logs.

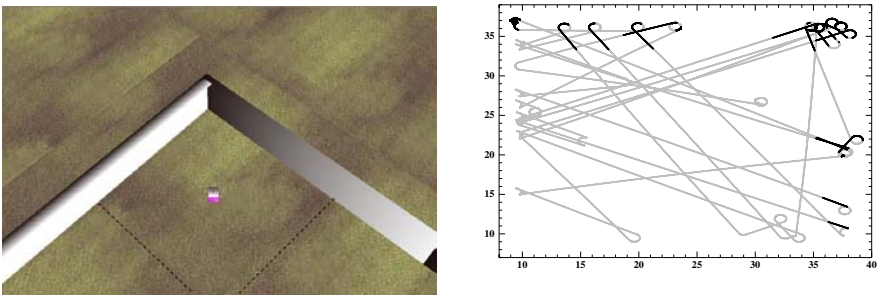
Two oracles will be constructed: one determines if the Wall is in View of the current robot, based on its 2D location; the other determines if the current robot can see another robot, based on the distance and angle between the two robots. It must be emphasized that these oracles are not, and cannot be, used in the robot life, for it does not know its location and does not know the location of the other robot either.

However, these oracles encapsulate a raw model of the sensors, the vision range of the robot; they can be acquired *in-silico* (reverse engineering the simulator) or *in-situ*, from actual experiments. A specific scenario is devised to learn each oracle. In each scenario, the sensorimotor stream of the robot is recorded and preprocessed. The robotic log stores for each robot and each time step the value of the two infrared sensors, the speed of the right and left motor, and the position of the robot in the arena[11].

The enriched events are used as training examples in order to learn the oracle primitives "I can see a wall" (respectively "I can see a robot"), as follows.

Seeing the walls. In this scenario, a robot is wandering around some empty arena, "bouncing" on the walls according to the randomized Braitenberg controller (Fig. 5.6). For each training sample, the label is *No* if both sensor values are the default value (the robot does not see anything) and *Yes* otherwise (if the robot sees anything,

[11] The orientation information is not reliably available from the simulator, in part due to the asymmetrical screw-drive system.



(a) Simulator view (with artificial limitation) (b) Result of learning along the trajectory

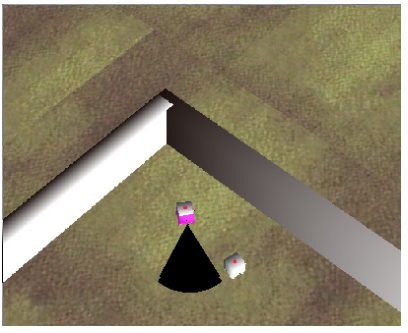
Fig. 5.6 Learning the *Wall-In-View* oracle. **(a)** A single robot moves in the artificially restricted arena; **(b)** shows the parts of its trajectory where it can see the wall, according to the hypothesis learned: the wall is visible from black parts of the trajectory, and not from the grey parts.

it is bound to be the wall). The input features are the coordinates and the orientation of the robot. These training examples are processed by a supervised learning algorithm (Sect. 5.1.4.2) to build a hypothesis, the Wall oracle. The *Wall-in-View* oracle is obviously specific to the current arena; it must be retrained for different arenas.

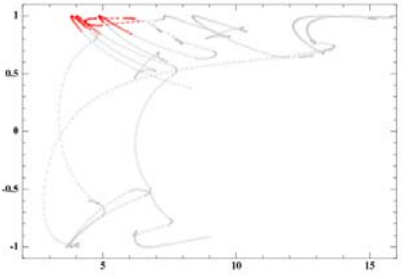
Seeing another robot. This scenario involves 2 robots. One is motionless while the other one performs prescribed trajectories, moving up and down at different distances around the other robot (see a sample trajectory in Fig. 5.7(c)). For each time step, the sensor values of the first robot are used to label the event: as for the Wall oracle above, the label is *No* if both sensors take the default value (the robot does not see anything) and *Yes* otherwise(if the motionless robot sees something it has to be the other robot). The input features are the distance between both robots and the angle θ between the orientation of the motionless robot and the azimuth of the moving robot (more precisely, $\cos(\theta)$). Figs. 5.7(c) and 5.7(d) show plots of the arena space, with the trajectory of the moving robot and the learned hypothesis (5.7(c)), and the error on the training set (5.7(d)). The learning task is here very easy, and the error is of only 2% on the training set and 4% on test sets involving other robots and trajectories.

Note that *Robot-In-View* oracle learned with this procedure actually describes the vision scope of the robot; it only depends on the robots, not on the arena. Hopefully, this oracle does not depend on the specific instance of robot used, but on the class of robot. However, whereas this can be reasonably argued within the simulator, it might not be exactly true with real robots[12].

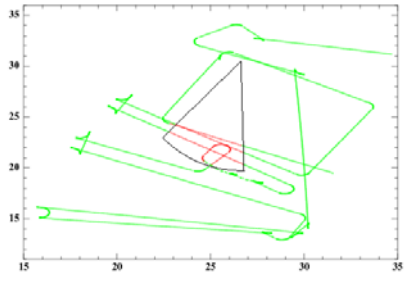
[12] Typically, when the hardware ages, possibly modifying the sensor performances, the vision scope might change as well; the oracles must then be retrained.



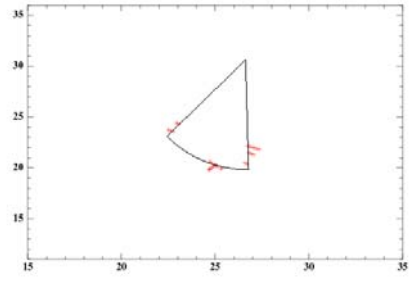
(a) Simulator view (vision cone added)



(b) Result of learning in (dist $\times$ cos) space: Black points are visible, grey points are not



(c) Trajectory of white robot and ground truth learned



(d) Error on the training set ($< 2\%$)

Fig. 5.7 Learning the *Robot-In-View* oracle. The top robot on plot (a) is motionless, while the other robot moves along the trajectory visible on plot (c), thanks to a manual control. The resulting vision cone can be visualized on the (distance $\times \cos(\theta)$ space on plot (b). It is clearly visible on the colour version of plot (c), but has been emphasized with the artificial vision cone of the motionless robot. The same cone also makes clear the errors made on the learning examples on plot (d) (similar errors, i.e. only at the border of the vision cone, happen on the test set).

These oracles constitute the first layer of the WOALA scheme; they use an information that the robot will never have in real-world experiments, and they will never be called again. They only support the automatic annotation of more complex logs, when several robots are moving in the same arena, meeting other robots and wall. These oracles can thus be viewed as robotic resources, akin an operational description of the physical system.

5.1.4.4 Learning Advanced Primitives

The proof of principle of the WOALA scheme concerns the distinction between viewing a wall, and viewing another robot. As already mentioned, acquiring a

primitive supporting this distinction constitutes a very first and key step in swarm and cellular robotics.

The difference between this learning task and the previous one, is that the available information in order to tell a wall from another robot only includes the sensorimotor information of the robot itself, contrasting with the use of the robot position in Sect. 5.1.4.3. On the one hand, the sensorimotor information is very poor; on the other hand, this is the only information actually available on-board for the robot.

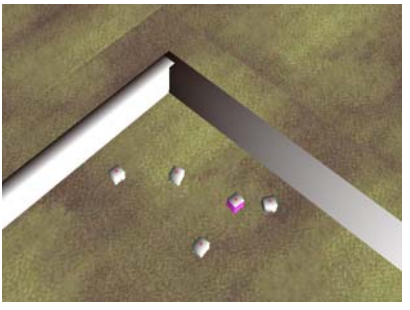
The different scenarios involve up to five robots wandering in wall-surrounded arenas. For each time step and each robot, one out of four labels holds:

- *I do not see anything* (the sensors take the default value). These events are filtered out.
- *I see a wall*. This label holds true when i) the robot sees something (one sensor value differs from the default value); ii) the Wall-In-View oracle is triggered; iii) the Robot-In-View oracle is not triggered.
- *I see a robot*. Symmetrically, this label holds true when i) the robot sees something; ii) the Robot-In-View oracle is triggered; iii) the Wall-In-View oracle is not triggered.
- *I see a complex scene*. This label holds true when i) the robot sees something (one sensor value differs from the default value); ii) both the Wall-In-View and the Robot-In-View oracles are triggered. In the first stages of the WOALA scheme, as the robot does not yet have the faculty to follow another robot, the *Complex* events are rare (about 2%), and they will be discarded in the following.
- *Aberration*. This label holds true when the robot sees something, although none of the Robot-In-View and Wall-In-View oracles are triggered. At the moment, these examples are filtered out: they correspond to some aberration of the sensors, or to oracle mistakes.

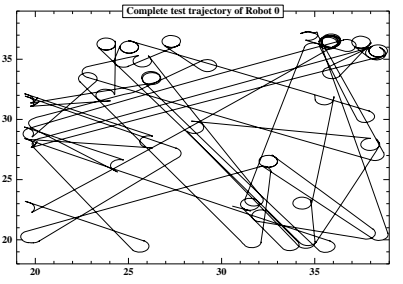
A supervised learning step of the WOALA scheme can then take place. A training example is described as the sensorimotor data collected at time $t, t-20, \ldots t-600$; the associated label corresponds to the label computed as above for time t. The goal is to endow the robot with an on-board virtual binary sensor (I see a Wall or I see a Robot), conditioned by the fact that something is visible (after the current value of its sensors). Further, the robot is assumed to have memorized its sensorimotor information within the last 600 time steps. Further work will be devoted to dimensionality reduction (Sect. 5.1.5).

5.1.4.5 First Results

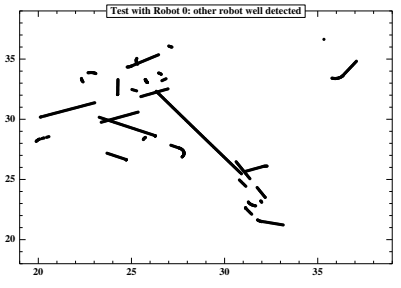
The experiment. In order to assess the WOALA scheme in a somehow realistic scenario, a restricted arena is used (a 20×20 square for a robot size of approximately 2, Fig. 5.8(a)); 5 robots are wandering in the arena, leading to a sufficient number of encounters, while nevertheless remaining tractable computational-wise. A typical trajectory for one robot is depicted on Fig. 5.8(b) (displaying the trajectories of more than one robot would not remain readable).



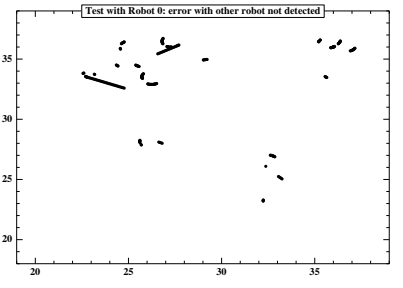
(a) The 5 robots in the arena



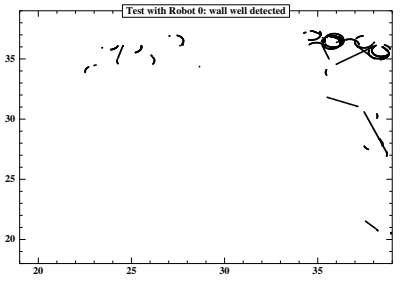
(b) The complete test trajectory of robot



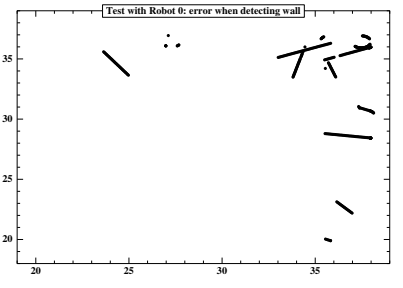
(c) Robot 0 correctly detects another robot



(d) Robot 0 fails to detect another robot



(e) Robot 0 correctly detects the wall



(f) Robot 0 fails to detect the wall

Fig. 5.8 Results of a complete WOALA loop. **(a)** shows the experimental setup – 5 robots in an empty arena. The robots wander around for some time, each trajectory being logged; **(b)** is an example of the trajectory of one robot; **(c)** and **(d)** represent respectively the correct predictions and the errors along the trajectory of plot (b) for the primitive "I see a robot" while plots **(e)** and **(f)** are similar displays for the primitive "I see a wall". Value of threshold τ_{global} was 0.001. See text for details.

The training set involves 10% of the events recorded for all 5 robots. The remaining events form the test set. 10 hypotheses are learned from 10 independent runs of the ROGER algorithm (Section 5.1.4.2), using the stochastic nature of the underlying optimization algorithm.

Table 5.1 Error rates when hypotheses learned on the trajectory of robot 0 are tested on trajectories of different robots in a different experiment.

τ_{global}	0.01		0.001	
Robot 0	21	9.5	24	2.7
Robot 1	23	12	26	5.6
Robot 2	23	11	25	5.3
Robot 3	23	12	26	4.6
Robot 4	24	12	27	5.9

The learned hypotheses are then tested on some test trajectory not involved of course during the training (such as the one shown on Fig. 5.8(b)), and all events (i.e. where the robot actually sees something according to its sensor values) are classified by the hypotheses thanks to a majority vote. The aggregated results are presented in Table 5.1, for 2 values of the global threshold τ_g for ensemble learning (see Sect. 5.1.4.2). The error cases are visually represented in Fig. 5.8.

5.1.5 Discussion and Perspectives

A first lesson to be drawn out of those results is that altogether, the learned hypotheses seem to be able to predict with 70 to 75% accuracy whether the robot is seeing a wall or another robot, based only on the very raw data of two infrared sensors. This is visually confirmed by the plots on Fig. 5.8. The WOALA scheme can be considered to be validated by those preliminary results, although many more experiments are required to assess the robustness of the approach.

A question under investigation regards the scalability of the approach as SYMBRION/REPLICATOR robots will typically use many more and much richer sensors (cameras, lasers) than the two infra-red sensors considered here. Interestingly, the presented log-based approach offers room for *off-line* dimensionality reduction (DR), regardless of whether the robot log has been acquired *in-silico* or *in-situ*. The state of the art provides quite a few effective approaches for (linear or non-linear) dimensionality reduction, ranging from Principal Component Analysis to Random Projection (Achlioptas, 2004), Isomap (Tenenbaum *et al.*, 2000) or LLE (Roweis & Saul, 2000). Furthermore, being an unsupervised method dimensionality reduction can be done off-line; the compression of the robot raw input can be optimized using extensive computational resources. The only requirement is that the resulting coding module be computationally frugal.

Another question regards the complexity of the WOALA decision. Whereas the full ensemble of hypotheses has been used in the experiments, on-line learning (Cesa-Bianchi & Lugosi, 2003) can be used to identify the most reliable hypotheses, discard the others and more generally enable lifelong learning through hypothesis update. The WOALA scheme would thus enable to build good enough primitives off-board, using remote computational resources; these primitives can be refined to some extent on-board, using the robot own computational resources.

The main weakness of the WOALA loop identified so far regards the early learning stage. Typically, labelling errors within the first learning stages can hardly be recovered, for it injects noise in all training evidence used to learn further concepts. While several revision steps have been performed so far, there is still room for improvement. A second weakness is that the learned primitives reflect the available evidence, which itself depends on the controller used to gather the robotic log. As new primitives are built and used to modify the controller, the distribution of the sensori-motor data is modified too.

A main perspective for further research regards the communication with the other robots, which is mandatory to the deployment of social intelligence. As discussed in Sec 5.1.2.2, the robot must be provided with a way to assess its behaviour. A preliminary step along this line is to make the robot self-aware through an anticipation module or a model of itself (Godzik *et al.*, 2004; Bongard *et al.*, 2006). Such a module indeed enables the robot to decide whether it is in normal mode (it correctly anticipates the effects of its actions) or not (the anticipation errors might be caused by e.g. a change in the environment, or a sensor or a motor failure); in the latter case, the robot can switch to a conservative mode (go to the nest).

The exploitation of the robotic log can however provide the robot with a much richer description of its state: clustering the sensorimotor data will define "states" (clusters). These states can be used to endow the robot with self-driven criteria using tools from Information Theory. On-going experiments show that maximizing the quantity of information of the robot trajectory enforces an exploratory behaviour (Delarboulas & Sebag, 2010).

Furthermore, the state information can be emitted and received by every robot, in the sense that any robot can compare its current state to the received state. How to build a proto-language from the sensori-motor states raises new challenges for collective intelligence.

5.2 Embodied, On-Line, On-Board Evolution for Autonomous Robotics

A.E. Eiben, Evert Haasdijk, Nicolas Bredeche

Artificial evolution plays an important role in several robotics projects. Most commonly, an evolutionary algorithm (EA) is used as a heuristic optimiser to solve some engineering problem, for instance an EA is used to find good robot controller. In these applications the human designers/experimenters orchestrate and manage the whole evolutionary problem solving process and incorporate the end result –that is, the (near-)optimal solution evolved by the EA– into the system as part of the deployment. During the operational period of the system the EA does not play any further role. In other words, the use of evolution is restricted to the pre-deployment stage.

Another, more challenging type of application of evolution is where it serves as the engine behind adaptation *during* (rather than before) the operational period,

without human intervention. In this section we elaborate on possible evolutionary approaches to this kind of applications, position these on a general feature map and test some of these set-ups experimentally to assess their feasibility.

The main contributions of this section can be summarised as follows:

- It provides a taxonomy of evolutionary systems encountered in related work. This taxonomy helps to identify the essence of particular approaches, to distinguish them from each other, and to position various options for a robotics project.
- It offers the first results of experiments aiming at the practical evaluation of (some of) these options.

5.2.1 Controllers, Genomes, Learning, and Evolution

In this subsection we elaborate on the fundamental notions of controllers, phenotypes, genotypes, learning, and evolution. We do not aim for universally valid definitions of these concepts (if such things are possible at all), rather at a consistent taxonomy and terminology reducing the chances of mis-communication.

An essential design decision when evolving robot controllers is to distinguish phenotypes and genotypes regarding the controllers. Simply put, this distinction means that:

- We perceive the controllers with all their structural and procedural complexity as phenotypes.
- We introduce a (typically structurally simpler) representation of the controllers as genotypes.
- We define a mapping from genotypes to phenotypes, that might be a simple mapping, or a complex transformation through a so-called developmental engine.

For example, a robot controller may consist of a group of artificial neural nets (ANNs) and a decision tree, where the decision tree specifies which ANN will be invoked to produce the robot's response in a given situation. This decision tree can be as simple as calling some ANN-1 when the robot is in stand-alone mode (not physically connected to other robots) and calling some ANN-2 when the robot is physically aggregated, i.e., connected to other robots. This complex controller, i.e., phenotype, could be represented by a simple genotype of two vectors, showing the weights of the hidden layer in ANN-1, respectively ANN-2. The two ovals and the link between them (including the developmental engine) in the middle of Fig. 5.9 shows this division.

A technical distinction between learning and evolution is now straightforward if we postulate that learning acts at phenotypic level, while evolution only affects the genotypes. As a consequence, we obtain two feedback-loops for adaptation, a learning loop and an evolutionary loop as shown in Fig. 5.9. There are a couple of things to be noted about this scheme. First, note that, just like the learning loop, the evolutionary feedback-loop includes the controllers, since the genotypes do not interact directly with the environment. Instead, the genotype determines the phenotype (controller) which in turn determines the robot behaviour. This behaviour in turn causes

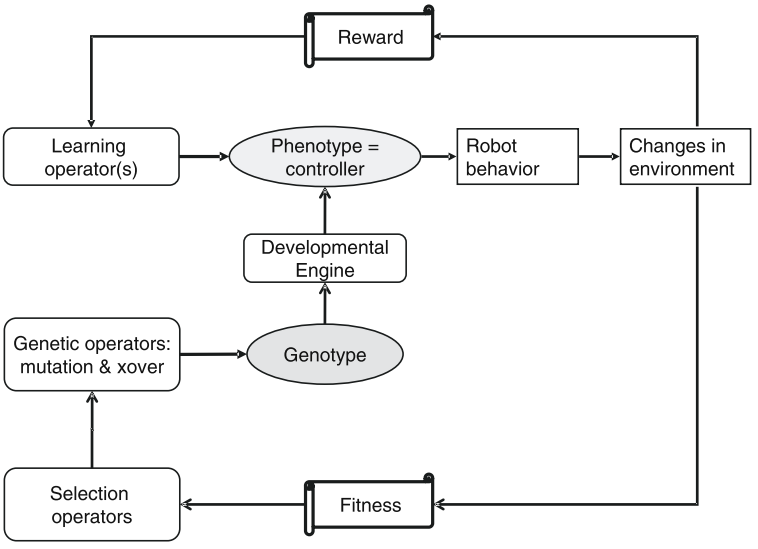


Fig. 5.9 General scheme of evolution and learning based on the genotype – phenotype distinction.

changes in the environment, including other robots. Second, both loops involve a utility measure, required to direct adaptation. For reasons of clarity we distinguish these measures also by name, using the term reward for learning and the term fitness for evolution. Third, please note that this simple diagram might become more complicated through the addition of more complex interactions. For instance, using Lamarckian operators we obtain learning on the genotype level, while an additional social learning mechanism can be naturally perceived as an evolutionary process on the phenotype level. Having noted all this, and keeping possible variations in mind, in essence we distinguish learning and evolution by their point of impact and the related time scale. Simply put, learning acts on phenotype level on the short term, within the lifetime of an individual (robot controller), while evolution works on genotypes on the long term, over consecutive generations.

5.2.2 Classification of Approaches to Evolving Robot Controllers

There are a number of features that allow us to position evolutionary robotics approaches. In (Schut *et al.*, 2009) we elaborated on the notion of situated evolution so that we could clarify similarities and differences between, for instance, regular GAs, spatially structured GAs, evolving ALife systems and evolutionary robotics. We now zoom in on evolutionary robotics and discuss different approaches by specifying a set of descriptive features and identifying a particular approach by the combination of features it belongs to. A previous attempt in this direction by Watson *et al.*

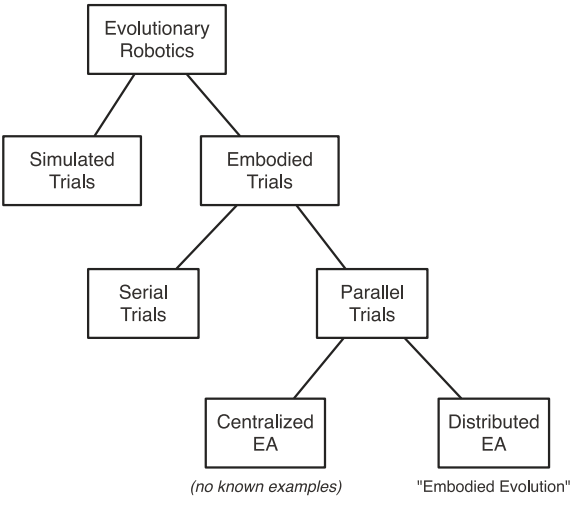


Fig. 5.10 Classification of evolutionary robotics approaches from (Watson *et al.*, 2002).

offers a classification scheme in (Watson *et al.*, 2002). This scheme is exhibited in Fig. 5.10. The figure shows that the primary distinction, i.e., the topmost junction in the graph, is based on embodiedness. This is also a key notion for us, since we are to apply evolution in real physical robots, hence in an embodied fashion. However, we want to go further in embodying evolution than doing the fitness evaluations through "embodied trials". We also want to embed the management and execution of evolutionary operators for selection and variation (i.e., mutation and crossover) in the robots. For a precise terminology we need to distinguish two essential components of an evolutionary process: the fitness evaluations, a.k.a. trials, on the one hand and the evolutionary operators on the other hand. Then we can also distinguish two basic types of embodied evolution: one where the fitness evaluations are embodied and one where the (management and execution of) evolutionary operators are embodied. The most common interpretation of the term embodied evolution coincides with the first case. Therefore, to prevent confusion, we will avoid using this term for the second case and will use the term on-board/intrinsic as introduced below.

Our classification scheme is based on a set of three features concerning the evolution of controllers from temporal, spatial, and procedural perspective. That is, we distinguish types of evolution considering when it happens, where it happens, and how it happens:

1. off-line or design time vs. on-line or run time (when),
2. on-board or intrinsic vs. off-board or extrinsic (where),
3. in an encapsulated or centralised vs. distributed manner (how).

Note, that we do not include embodiedness (of fitness evaluations) in this classification scheme. The reason is that the system we have in mind is one with real robots, where fitness evaluation always happens in reality. In other words, our whole scheme falls in the category of embodied evolution in the terminology after Watson *et al*.

In *off-line* evolution, the evolutionary development[13] of robot controllers takes place *before* the robots start their "real" operation period. *On-line* evolution is the opposite in that the evolutionary development of robot controllers takes place *during* the "real" operation period of the robots (although off-line evolution might precede on-line evolution as an educated initialisation procedure) and is an ever-continuing process. Obviously, the distinction between these two options lies in the release moment when the evolved controllers are deployed in the robots. If the evolutionary operators are no longer applied after the release moment and the controllers remain fixed (or only change by other mechanisms), we are dealing with the off-line case, otherwise we have on-line evolution.

From the spatial perspective, we distinguish the *on-board* or *intrinsic* case where the evolutionary operators such as selection, crossover, and mutation are performed exclusively inside the actual robot hardware, from the *off-board* or *extrinsic* case, where they are performed with the help of external equipment outside the robots. Such external equipment could be a computer, interfaced with the robots, that plays the role of "puppet master" in an on-line evolutionary process: based on fitness information it collects from the robots (embodied trials!) it manages the evolutionary operators for selection and variation and injects newly produced controllers into the existing robot bodies. If we view a system such as this in terms of parallel EAs –where there are many corresponding considerations– we would describe it as a master-slave parallel EA with the slaves calculating fitness and the master orchestrating evolution. Here we can recall our elaboration on embodiedness in the beginning of this section. As we observed there, it would be formally correct to describe what we call on-board or intrinsic evolution as embodied evolution because the evolutionary operators are embodied in the robots. The reason to choose other terms here is twofold. First, the usual terminology associates embodied evolution with embodied trials, which is a completely different thing. Second, introducing new terms here facilitates precise phrasing: embodied evolution means that fitness evaluations (trials) are done in real-life by the robots, while on-board or intrinsic evolution means that the evolutionary operators executed by the robots.

Last, but not least, we consider how the evolutionary operators are managed. First, we distinguish the *distributed* approach, where each robot carries one genotype and is controlled by the corresponding phenotype. Robots can reproduce autonomously and asynchronously to create offspring controllers by recombination and/or mutation. Here, the iterative improvements (optimisation) of controllers result from the evolutionary process that emerges from the interactions between the robots. In terms of parallel EAs, such a distributed system is analogous to a cellular parallel EA. This approach is complemented by the *encapsulated* or *centralised* approach: each robot has a completely autonomous EA implemented on-board, maintaining a population of genotypes inside itself. These EAs can be different for different robots and are executed in a traditional, centralised manner

[13] Development is meant here in the engineering sense, the "making of" or "building of" controllers, rather than the biological, embryo-genetic sense as "creating it from an embryo". We use the term development to hint at the iterative nature of this process.

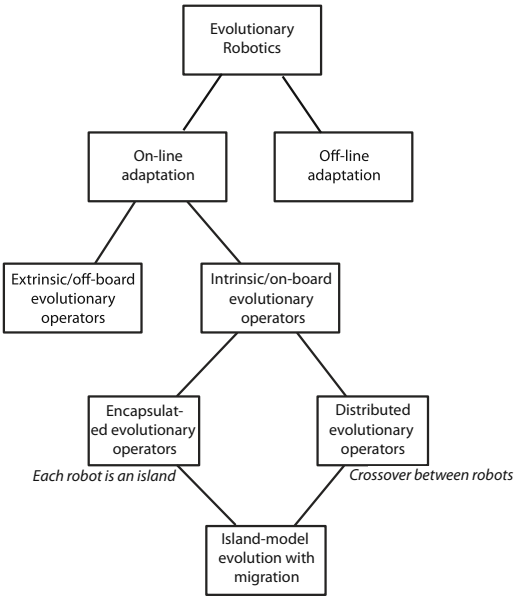


Fig. 5.11 Proposed (partial) classification of evolutionary robotics approaches.

locally, inside each robot. This is typically done by a time-sharing system, where one genotype from the inner population is activated (i.e., decoded into a phenotype controller) at a time and is used for a while to gather feedback on its quality. Here, the iterative improvements (optimisation) of controllers are the result of the EAs running in parallel inside the individual robots independently. In terms of parallel EAs, such a distributed system is analogous to an island-model parallel EA (without migration). Observe, that both the encapsulated and the distributed approaches yield a population of heterogeneous robot controllers. Furthermore, it is important to note that we distinguish distributed and centralised control of the EA, not of not the robots per se: they perform their tasks autonomously in all cases. Finally, a remark on the term centralised and encapsulated as defined here. To some extent, we use them as synonyms, both being the counterpart of the distributed approach. Strictly speaking the adjective "centralised" would already suffice, but we also introduce "encapsulated" to emphasise the fact that a (centralised) evolutionary algorithm is running entirely inside a robot.

Fig. 5.11 shows a classification graph along the lines described here. At first glance, this might seem at odds with the one in Fig. 5.10. However, the distinction between on- and off-line renders the two dichotomies based on "trials" (simulated vs embodied and serial vs parallel) superfluous: on-line adaptation only makes sense in real robots (although for experimental purposes, the whole system may be simulated): adaptation takes place as the robots go about their tasks and performance evaluation is inherently parallel across robots. Thus, the distinction between extrinsic and intrinsic EA operators matches the one between centralised and distributed

EA and the further distinctions under the intrinsic case could be seen as a refinement of the "Distributed EA" leaf in Fig. 5.10.

5.2.3 The Classical Off-Line Approach Based on a Master EA

Using the classification scheme based on the three features we have discussed it is possible to characterise existing approaches to evolutionary robotics (Floreano *et al.*, 2008a; Nolfi & Floreano, 2000a). The usual approach to ER is to use a conventional EA for finding good controllers in a fashion that can be identified as

- off-line,
- extrinsic (off-board)
- centralised (encapsulated in an external computer).

Fig. 5.12 illustrates this approach. Note, that the arrow from the external computer to the robot represents the final deployment of the best found controller after the evolutionary search is finished. The figure does not show how the fitness evaluations are done during the evolutionary search. In other words, this figure covers the possibilities of evaluations in simulation as well as in real-life, i.e., in an embodied fashion.

The on-line evolutionary system we have in mind is radically different from this approach in that adaptation of the robots never stops. From our perspective, this means that evolution is being performed on-the-fly, possibly combined with other adaptive processes, such as individual learning or social learning. In the next section we discuss a number possible systems for on-line evolution.

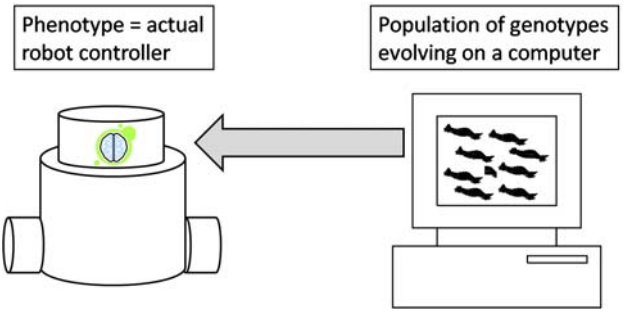


Fig. 5.12 The common off-line approach to evolutionary robotics, where the population of genotypes is evolved on a computer that executes the evolutionary operators (variation and selection), managed in a centralised fashion. As for fitness evaluation, the computer can invoke a simulator or send the genotype to be evaluated to a robot to test it (embodied evaluation). Evaluation of genotypes can happen in parallel. At the end of the evolutionary process the best genotype is deployed in the robot(s).

5.2.4 On-Line Approaches

5.2.4.1 Encapsulating the EA in the Robots

This option amounts to implementing the EA to the robots to run it inside the robots while they are operating. Obviously, this implies a whole population of genotypes being hosted in one robot, while the robot can have only one controller at a time. This means that at any given time only one of the genotypes is activated, i.e., decoded into a controller. Fig. 5.13 illustrates this matter.

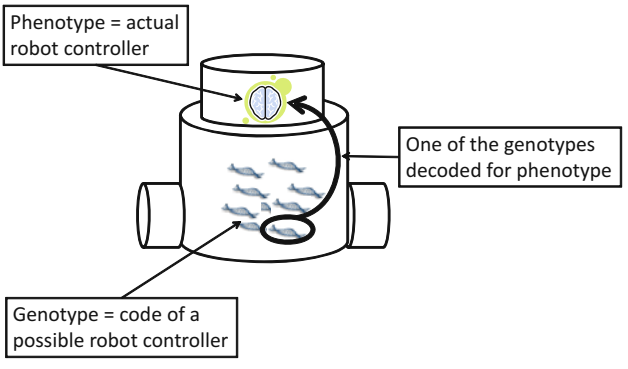


Fig. 5.13 Encapsulated evolution illustrated in one robot hosting an evolving population of genotypes. At any given time one of these genotypes is activated, i.e., decoded into a controller. Execution of evolutionary operators (variation and selection) takes places inside the robot, on-board, managed in a centralised fashion. Fitness evaluations are typically performed by activating the genotypes one by one through a time-sharing system and using them for a while.

We will use the term encapsulated EA to designate this approach[14]. This approach is seldomly used with only two examples we know of (Nehmzow, 2002; Usui & Arita, 2003). Such a system, illustrated in Fig. 5.14, can be described as

- on-line,
- on-board (intrinsic),
- encapsulated (centralised).

The most natural option, matching the on-line character of this set-up, is in vivo fitness evaluation of genotypes by transforming them into phenotypes and using them to control the given robot for a while. After this evaluation period, another genotype can be transformed into phenotype/controller to undergo its own evaluation. Thus, all robots run their own EA on-the-fly, so we have a number of parallel evolutionary processes running independently as shown in Fig. 5.14.

[14] In (Nehmzow, 2002) this is called "embedded", but we feel that embodied and embedded can be easily confused, therefore choose "encapsulated".

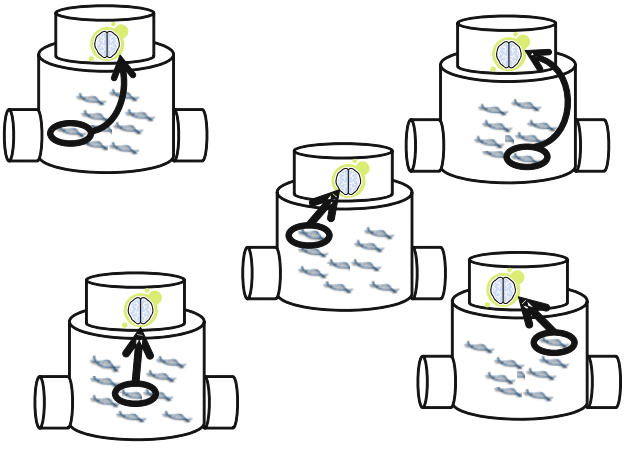


Fig. 5.14 Encapsulated evolution in a group of robots, where each robot is running a (centralised) evolutionary algorithm on-board independently. The evolutionary process does not require communication and interaction between robots.

5.2.4.2 Distributed Evolution

Our next example illustrates a set-up that is similar to ALife-like evolution with natural selection and natural reproduction (Eiben *et al.*, 2007). The difference with such ALife systems is caused by the practical constraint that robot bodies do not multiply. This implies that we have a fixed number of placeholders for controllers, the bodies, hence we cannot add a new controller to the population without removing an old one. Death of a controller without immediate replacement is in principle possible, but would amount to a waste of resources (inactive robot), thus we expect a mechanism to prevent this. This all means that we have a "half-natural" reproduction, where reproduction and survivor selection are not independent, but mating is autonomous and asynchronous. Using our feature set this approach can be described as

- on-line,
- on-board (intrinsic),
- distributed.

Obviously, a decentralised system lacks a global puppet master orchestrating the process of evolution. Rather, the evolutionary process is a result of activities of the individual robots. In other words, all evolutionary operators for selection and reproduction are managed autonomously. Fig. 5.15 illustrates this type of on-line evolution.

5.2.4.3 Distributed and Encapsulated Evolution

Obviously, it is possible to combine the mechanisms of encapsulated and distributed evolution. From the encapsulated EA perspective, this means extending the

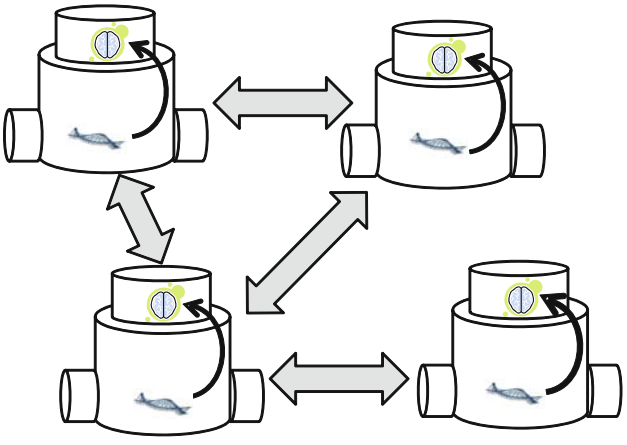


Fig. 5.15 Distributed on-board evolution, where each robot carries its own single genotype and evolution emerges from the reproductive interactions between robots. These interactions are indicated by the grey arrows.

system with migration of genotypes between robots. In this case we obtain interacting evolutionary processes, similar to the island model with migration for parallel evolutionary algorithms. This option is depicted in Fig. 5.16. Using our feature set this approach can be described as

- on-line,
- on-board (intrinsic),
- distributed *and* encapsulated.

5.2.4.4 Master EA Orchestrating On-Line Evolution

The basis of this approach is the existence of a central authority to manage the evolutionary operators for selection and reproduction, while running in the on-line mode. Technically, this means that the given group of robots acts as a group of slaves (purely in terms of the EA). In this (heterogeneous) group each robot carries one genotype and sends fitness information to the master. Then it is the master who decides –using the global information it possesses– which robot controllers are to be recombined and/or mutated and which ones should be replaced with newly created controllers. The creation of new genotypes can take place inside this computer and the result deployed in the robots whose controller is selected for replacement. The genotype sent by the master is decoded/activated into a phenotype, i.e., into a working controller. From the perspective of the master this means that fitness evaluations can be done in real life and in parallel. If the group size of the robots equals the population size within the EA then the whole population can be evaluated simultaneously. From the perspective of the robots this means that they form a heterogeneous group and their controllers are repeatedly replaced by new ones – that might be better or worse than the one used before.

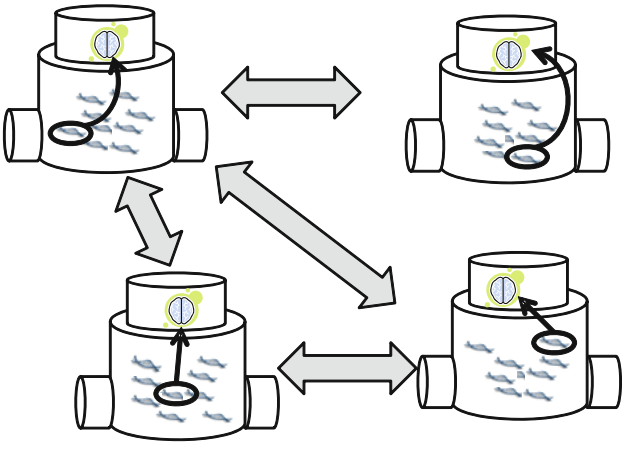


Fig. 5.16 On-board evolution where each robot is running an evolutionary algorithm inside and genotypes can migrate between robots. Execution of evolutionary operators takes places inside the individual robots, but communication and interaction between robots is required for the migration of genotypes. These interactions are indicated by the grey arrows.

In terms of the classification scheme of Watson *et al.* in Fig. 5.10, this option belongs to the leaf on the path Embodied Trials – Parallel Trials – Centralised EA, with no known examples. The describing properties of this system are:

- on-line,
- extrinsic (off-board),
- centralised (encapsulated in an external computer, not in the robots).

A possible argument for using such a system is that the global information of the master and its ability to fully control selection and reproduction makes it easier to evolve good controllers than using a decentralised architecture. This type of evolution is illustrated in Fig. 5.17.

5.2.5 Testing Encapsulated Evolutionary Approaches

In this section we report on the first experiments with encapsulated evolution.

5.2.5.1 The $(\mu + 1)$-ONLINE Evolutionary Algorithm

These experiments validate an encapsulated EA that is based on the classical $(\mu + 1)$ evolution strategy (Schwefel, 1995), where a population of μ individuals is maintained within each robot.[15] Here, each individual is the genotypic code of a robot

[15] Note that the population size μ within the EA should not be confused with the group size, i.e., the number of robots in the arena.

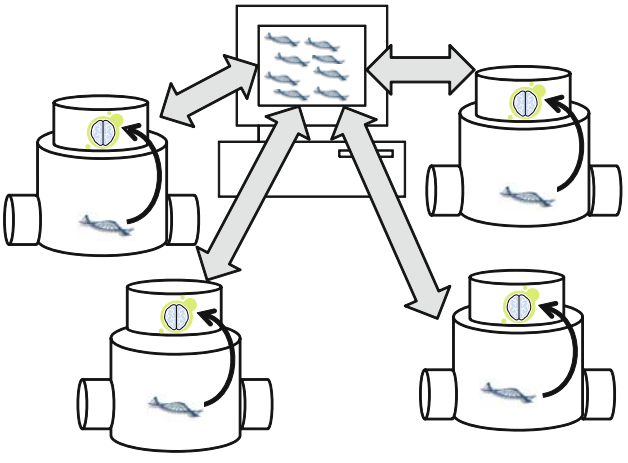


Fig. 5.17 Off-board on-line evolution where each robot carries one genotype and evaluates its fitness by using the controller it encodes. Evolution is managed by an external master computer that executes all evolutionary operators for selection and reproduction. Communication between the robots and the master is required for transmitting genotypes and fitness information. These interactions are indicated by the grey arrows in the figure.

controller that is in the form of an artificial neural net (ANN). This ANN is a perceptron with a hyperbolic tangent activation function using 9 input nodes (8 sensor inputs and a bias node), no hidden nodes and 2 output nodes (the left and right motor values), 18 weights in total. These 18 weights are to be evolved, therefore, the evolutionary algorithm will use the obvious representation of real-valued vectors of length 18 for the genomes. For the first experiments we decide to set the population size $\mu = 1$, restricting ourselves to a $(1 + 1)$ evolution strategy and consequently we omit recombination. We use a straightforward Gaussian mutation, adding values from a distribution $\mathcal{N}(0, \sigma)$ to each x_i in the genotype $\bar{x}$. This simple scheme defines the core of our EA, but it is not sufficient to cope with a number of issues in our particular application. Therefore, we extend this basic scheme with a number of advanced features, described below.

1. **Adapting σ values.** A singleton population is inherently very sensitive to premature convergence to a local optimum. To overcome this problem, we augment our EA with a mechanism that varies the mutation step-size σ on the fly, switching back and forth between local and global search, depending on the course of the search. In particular, σ is set to a pre-defined minimum to promote local search whenever a new genome is created. Then, σ gradually increases up to a maximum value (i.e., the search shifts towards global search) as long as the no improvements are found to the best genome found so far (the so-called champion, stored in the robot's archive). If local search leads to improvements, σ remains low, thus favouring local search. Otherwise the increasing σ values will move the search into new regions in the search space.

2. **Recovery period.** Because we use *in vivo* fitness evaluation, a new genome needs to be "activated" to be evaluated: it has to be decoded into a controller and take control of the robot for a while. One of the essential design decisions is to avoid any human intervention during evolution, such as repositioning the robot before evaluating a new genome. Consequently, a new controller will start where the previous one finished, implying the danger of being penalised for bad behaviour of its predecessor that may have manoeuvred itself into an impossibly tight corner. To cope with this effect, we introduce a *recoveryTime*, during which robot behaviour is not taken into account for the fitness value computation. This favours genomes that are efficient at both getting out of trouble during the recovery phase and displaying efficient behaviour during the evaluation phase.
3. **Re-evaluation.** The evaluation of a genome is very noisy because the initial conditions for the genomes vary considerably: an evaluation must start at the final location of the previous evaluation, leading to very dissimilar evaluation conditions from one genome to another. For any given genome this implies that the measurement of its fitness, during the evaluation period, may be misleading, simply because of the lucky/unlucky starting conditions. To cope with such noise, we re-evaluate the champion (i.e., current best) genome with a probability $P_{re-eavulate}$. This is, in effect, a resampling strategy as advocated by Beyer to deal with noisy fitness evaluations (Beyer, 2000). As a consequence, the robots needs to share its time between producing and evaluating new genomes and re-evaluating old ones. The fitness value that results from this re-evaluation could be used to refine a calculation of the average fitness of the given genome. However, we choose to overwrite the previous value instead. This may seem counterintuitive, but we argue that this works as a bias towards genomes with low variance in their performance. This makes sense as we prefer controllers with robust behaviour. It does, however, entail an intrinsic drawback as good genomes may be replaced by inferior, but lucky genomes in favourable but specific conditions. Then again, a lucky genome which is not good on average will not survive re-evaluation, avoiding the adaptive process getting stuck with a bad genome.

The resulting method is called the $(\mu + 1)$-ONLINE evolutionary algorithm; its pseudo code is shown in Algorithm 5.

5.2.5.2 $(1 + 1)$ Encapsulated Evolution in a Hybrid Set-Up

These results have been published more extensively in (Bredeche *et al.*, 2009), therefore here we only give a brief summary of the method and the most important outcomes.

In the first series of experiments we tested the $(\mu + 1)$-ONLINE algorithm in a hybrid set-up that features actual robotic hardware, a Cortex M3 board with 256kb memory. This controls a simulated autonomous e-puck in a Player/Stage environment. After N time-steps, the evaluation of the current controller is complete and the controller parameters are replaced with values from a new genome, which is

Algorithm 5. The $(\mu + 1)$-ONLINE evolutionary algorithm.

```
   // Initialisation
1  for i = 1 to μ do
2      population[i] = CreateRandomGenome
3      population[i].Fitness = Fitness_min
4  end
5  for evaluation = 1 to N do
6      Parent = SelectRandom(population)
7      if random() < P_re-evaluate then
           // Don't create offspring, but re-evaluate selected
              parent itself
           // Get out of bad situations due to previous
              evaluation
8          Recover(Parent)
           // Combination depends on re-evaluation strategy:
           // overwrite, average or exponential moving avg.
9          Parent.Fitness = Combine(Parent.Fitness, RunAndEvaluate(Parent))
10         Sort(population)
11     end
12     else
           // Create offspring and evaluate that as challenger
13         Challenger = Mutate(Parent, Parent.σ)
           // Get out of bad situations due to previous
              evaluation
14         Recover(Challenger)
15         Challenger.Fitness = RunAndEvaluate(Challenger)
16         if Challenger.Fitness > population[μ].Fitness then
17             population[μ] = Challenger
18             population[μ].Fitness = Challenger.Fitness
19             population[μ].σ = σ_min
20             Sort(population)
21         end
22         else
23             Parent.σ = Parent.σ · 2
24         end
25     end
26 end
```

evaluated *from the location the previous controller left it in*. This means that *no* human intervention is *ever* needed. We run the experiment 12 times.

Fig. 5.18 illustrates the experimental set-up, with a Cortex board connected to the computer running Player/Stage. The simulated robot is modelled after an ePuck mobile robot with two wheels and eight proximity sensors. The maze environment used in our experiment is as shown in this figure.

For each run of the experiment, the robot starts with a random genome and a random seed. The fitness function promotes exploration and is inspired by a

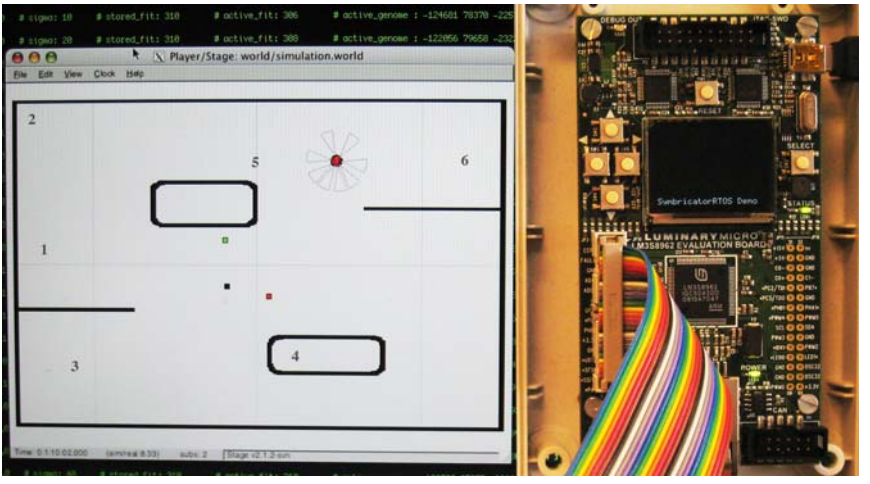


Fig. 5.18 The experimental setup: the Cortex board connected to Player/Stage. The numbers in the player-stage arena indicate the starting positions for the validation trials.

Table 5.2 Experiment description table for the $(1+1)$-ONLINE evolution tests

Experiment details	
Task	fast forward
Arena	see Fig 5.18
Robot group size	1
Simulation length	1000 time steps
Controller details	
ANN type	perceptron
Input nodes	9 (8 sensory inputs and 1 bias node)
Output nodes	2 (left and right motor values)
Evolution details	
Representation	real valued vectors with $-4 \leq x_i \leq 4$
Chromosome length L	18
Fitness	See equation 5.1
Recovery time	30 time steps
Evaluation time	30 time steps
$P_{reevaluate}$	0.2
Population size μ	1
Mutation	Gaussian $N(0,\sigma)$ with adaptive σ values, $\sigma_{initial} = 1$
Crossover	n/a
Parent selection	n/a
Survivor selection	replace when better

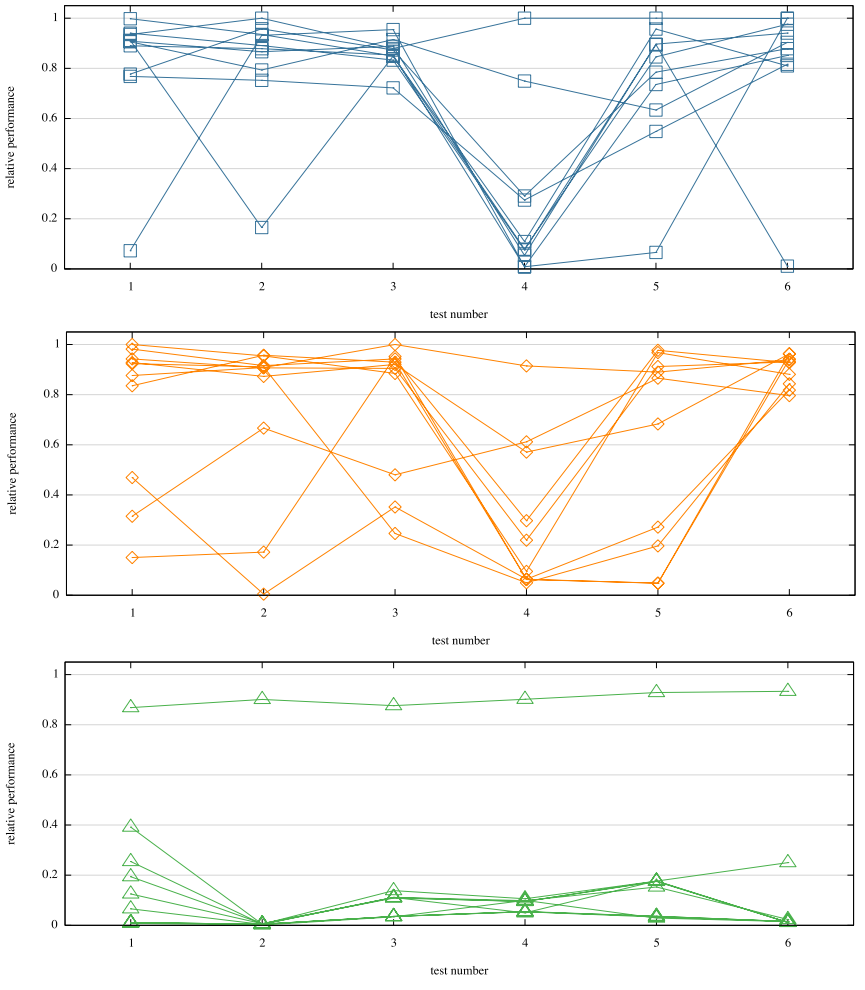


Fig. 5.19 Performance on validation scenarios for various re-evaluation schemes. Top: overwrite-last-fitness scheme; Middle: average-fitness scheme; Down: no re-evaluation scheme. X-axis shows the results on the six different validation setup (see Fig. 5.18), y-axis shows normalised fitness performance for each run. For a given genome, results in the six validation set-ups are joined together with a line.

classic one, described in (Nolfi & Floreano, 2000a) which favours robots that are fast and go straight-ahead, which is of course in contradiction with a constrained environment, implying a trade-off between translational speed and obstacle avoidance. Equation 5.1 describes the fitness calculation:

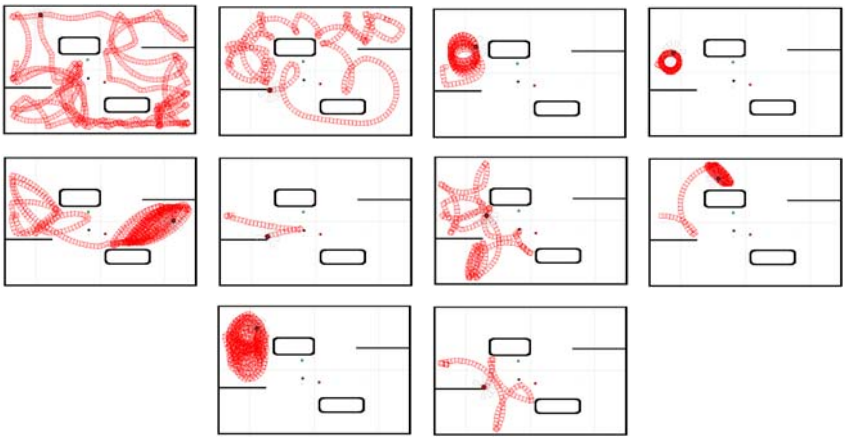


Fig. 5.20 Traces for the ten best controllers (using fitness replacement after re-evaluation).

$$fitness = \sum_{t=0}^{evalTime} (speed_{translational} \cdot (1 - speed_{rotational}) \cdot (1 - minSensorValue)) \tag{5.1}$$

The overview of experimental details is given in Table 5.2.

To provide an indication of the true performance and reusability of the best individuals found by $(\mu + 1)$-ONLINE evolution, a hall-of-fame is computed during the course of evolution from the champions of all runs. The 10 best genomes from the hall-of-fame are validated by running each from six initial positions in the environment, indicated in Fig. 5.18. Starting from each of these positions, the genomes are evaluated for ten times the number of steps used for evaluation during evolution. Note, that one of the validation starting positions has certainly never been visited during development (test no.4, within a small enclosed area) and provides an extreme test case in a very constrained environment. This decomposition into an evolution (development) phase and a post-experiment testing phase is similar to the learning and testing phases commonly seen in Machine Learning and does not imply a deployment phase as in traditional, off-line evolutionary robotics approaches.

We conducted a series of twelve independent experiments $(\mu + 1)$-ONLINE evolution, with parameters set as stated above. Each experiment started with a different random controller (with very poor behaviour indeed) and a different random seed. The experiments ran for 500 evaluations and displayed different overall fitness dynamics with very similar patterns. In all our experiments, we saw a similar pattern of initial random search characterised by many different genomes with poor fitness; then, local search characterised by subsequent genomes with increasing fitness until a robust genome is found that survives re-evaluation for some time and then a switch to another region that yields good results or towards an inferior genome that got lucky (almost a restart, in effect).

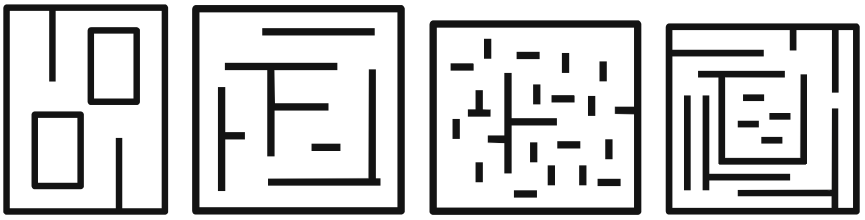


Fig. 5.21 The four arenas used in the second series of experiments; **(a)**: arena 1, **(b)**: arena 2, **(c)**: arena 3, **(d)**: arena 4.

During the course of the experiments a hall-of-fame was maintained for further validation of the best genomes. Fig. 5.19 shows the results of the validation of the hall-of-fame for three different schemes for re-evaluation: overwrite-last-fitness, where the champion's fitness is overwritten after every re-evaluation, average-fitness, where the fitness is the average of all re-evaluations and a scheme where there is no re-evaluation at all. This allows us to assess two things: whether high ranking genomes in the hall-of-fame are also efficient in a new set-up and whether

Table 5.3 Experiment description table for the $(\mu + 1)$-ONLINE encapsulated evolution tests

Experiment details	
Task	fast forward
Arena	4 different arenas, see Fig 5.21
Robot group size	5
Simulation length	1000 time steps
Controller details	
ANN type	perceptron
Input nodes	9 (8 sensory inputs and 1 bias node)
Output nodes	2 (left and right motor values)
Evolution details	
Representation	real valued vectors with $-4 \leq x_i \leq 4$
Chromosome length L	18
Fitness	See equation 5.1
Recovery time	30 time steps
Evaluation time	30 time steps
$P_{reevaluate}$	0.2
Population size μ	1, 3, 9, 13
Mutation	Gaussian $N(0, \sigma)$ with adaptive σ values, $\sigma_{initial} = 1$
Crossover	none
Parent selection	random
Survivor selection	replace worst when better

the "overwrite fitness" re-evaluation scheme is relevant. The y-axis shows the normalised performance: the best of all individuals for a scenario is set to 1.0, the performance of the other individuals is scaled accordingly. For each scenario (arranged along the x-axis), the graphs show a mark for each individual from the hall-of-fame. All results for a given genotype are linked together with a line.

The graphs clearly show that re-evaluation improves performance substantially; from the ten best solutions without re-evaluation, only a single one performs at a level comparable to that of the ones with re-evaluation. It is harder to distinguish between the two algorithm variants using re-evaluation: averaging the fitness measurements for the genome in question or overwriting the archived fitness value with the last measurement. On the one hand, the spread of performance *seems* greater for the case with averaging fitness than it does for overwriting fitness, which would endorse the reasoning that overwriting after re-evaluation promotes individuals with high average fitness and low standard deviation. On the other hand, however, the nature of real world experiments have a negative impact on the amount of data available for statistically sound comparison of re-evaluation strategies, as is often the case with real hardware, and keep from formulating a statistically sound comparison.

Further analysis of the ten best individuals with the overwrite-fitness re-evaluation scheme shows that the controllers actually display different kinds of behaviour –all good, robust, but different wall avoidance and/or open environment exploration strategies, ranging from cautious long turns (reducing the probability of encountering walls) to exploratory straight lines (improved fitness but more walls to deal with). Fig. 5.20 illustrates this by showing the pathways of these individuals, starting from an initial position on the left of the environment. This reflects the genotypic diversity observed in the hall-of-fame and hints at the algorithm's capability to produce very different strategies with similar fitness.

5.2.5.3 $(\mu+1)$-ONLINE Evolution in a Simulated Set-Up

The second series of experiments increases the population size μ beyond 1. These experiments are performed in a pure simulation environment (Delta3D-based) utilising four different arenas shown in Fig. 5.21. An additional difference with the first series of experiments is that here we use a group of five robots that are active in the given arena simultaneously. The robots obviously pose additional, moving obstacles for each other, but do not otherwise interact. For each arena and value of μ, we conducted 10 trials. To keep the experimental setup as close as possible to the previous one, we decided not to use crossover and not to use fitness-based parent selection either, because none of them was possible in the $(1+1)$ case. In this way we can study the effect of population size in isolation.

The $(\mu+1)$-ONLINE algorithm allows individual robots to maintain a larger population. We hypothesise that using larger populations has a positive effect on the quality of the evolved controllers and test this hypothesis by experiments using $\mu = 1,3,9,13$ in each of the four arenas. The overview of experimental details is given in Table 5.3.

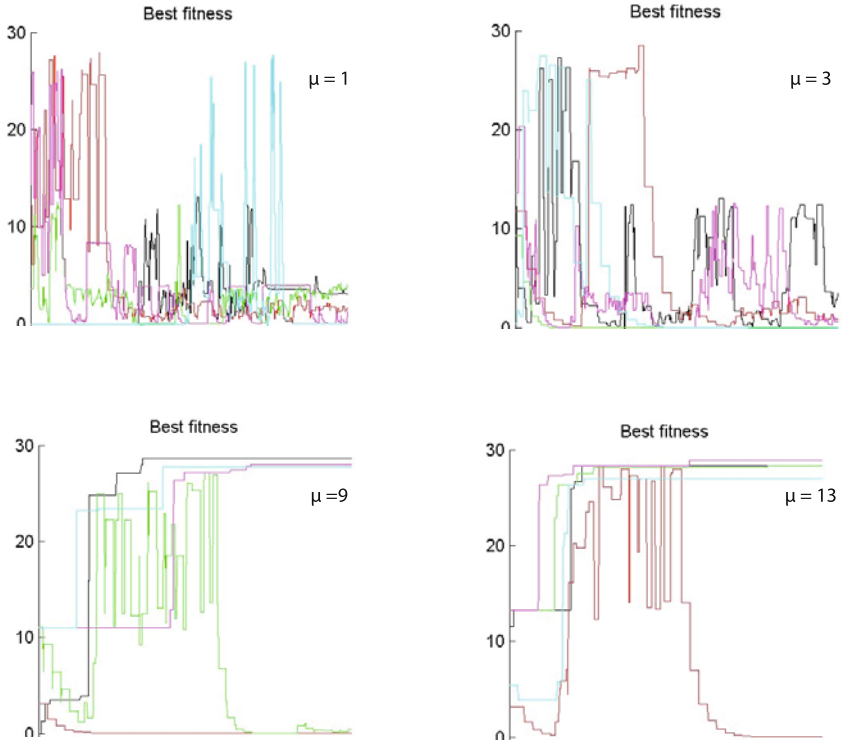


Fig. 5.22 Typical runs for $\mu = 1, 3, 9, 13$. The y-axis shows the performance of the current champion, the y-axis the number of generations . Each graph shows five plots; one for each of the five robots in the arena.

Fig. 5.22 shows the development of champion performance as evolution progresses for typical runs with varying values of μ. We clearly see that lower values of μ display considerable drops in champion fitness–much more so than large values. Such drops are, as noted in Sect. 5.2.5.2 the effect of the inherently noisy fitness calculation when an actually quite poorly performing genome is evaluated as having a high fitness due only to auspicious circumstances. Obviously, small populations are more susceptible to removing a good individual after evaluating such a lucky challenger. For example, for $\mu = 10$, evolution would have to encounter 10 lucky challengers before actually removing the champion, but with $\mu = 1$, the champion is dropped immediately.

By the same token, the champion fitness at the end of the runs as shown in Fig. 5.23 is better for larger values of μ: good individuals are more easily forgotten for low values of μ, thus more runs will end with low champion fitness. The peak performance (best champion ever), however, does not vary with μ.

To analyse actual robot performance, we show the average fitness including challengers and re-evaluations over the last 20 evaluations in Fig. 5.24. Here, we see

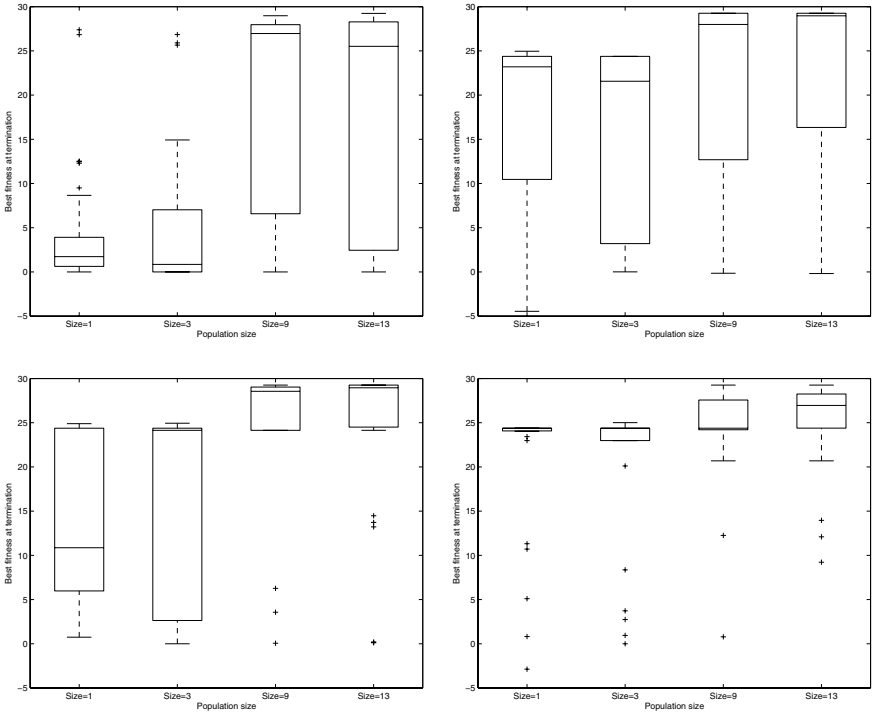


Fig. 5.23 The effect of increasing the population size μ on champion fitness. Each box summarises champion performance in 10 runs with μ set to 1, 3, 9 or 13; the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. Top-left: arena 1, top-right-arena 2, lower-left: arena 3, lower-right: arena 4.

that the actual performance does not increase with μ–in fact, it is worse. This may be explained by the fact that, just as it takes time to forget a good champion, it takes time to forget lucky but actually bad genomes that made it into the population. We expect to be able to mitigate this effect by introducing non-random parent selection, reducing the likelihood of selecting poorly performing genomes from the population. Also, increasing the re-evaluation rate (fixed at 0.02 for these experiments) is likely to reduce the time needed to recognise poorly performing genomes.

5.2.6 Conclusions and Future Work

In this section we presented a new taxonomy to classify evolutionary robotics applications, based on three main features belonging to the "when", "where", and "how" dimensions. We deliberately focused on systems with *in vivo* (embodied) fitness evaluations, where the evolutionary algorithm is running on-line without

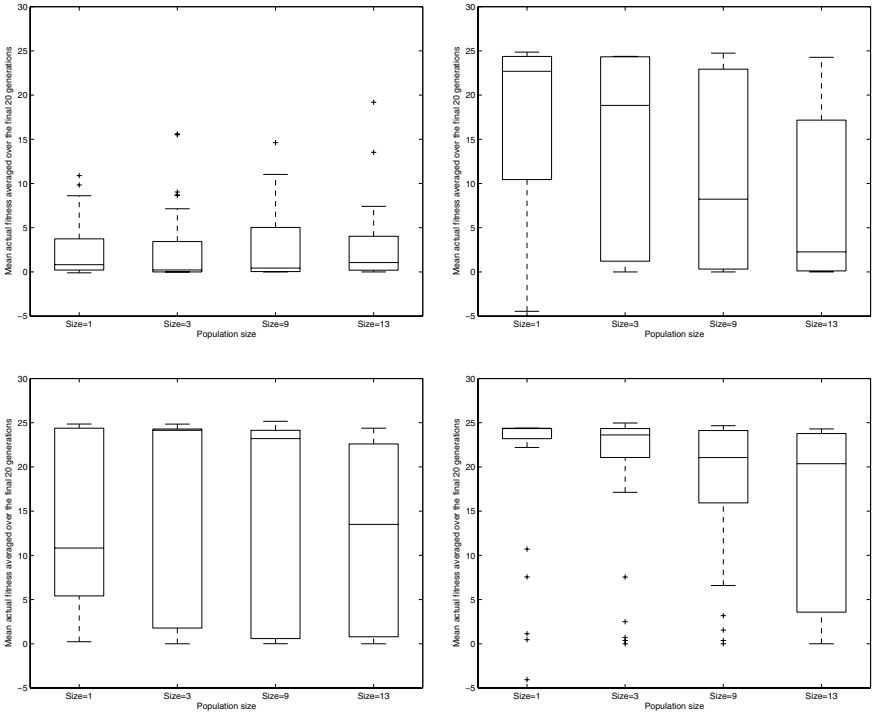


Fig. 5.24 The effect of increasing the population size μ on actual fitness. Each box summarises actual performance in 10 runs with μ set to 1, 3, 9 or 13; the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. Top-left: arena 1, top-right-arena 2, lower-left: arena 3, lower-right: arena 4.

human intervention, and the evolutionary operators are managed on-board either in a centralised/encapsulated or distributed or mixed fashion.

We also reported on the first experiments with on-line, on-board, encapsulated evolution using different population sizes (not to be confused with the number of robots) by means of a feasibility study within the projects. The results indicate that the approach is feasible even in a most simple setup, with random parent selection and no crossover, and show that increasing the population size improves the quality of evolved controllers. Currently we are conducting experiments with advanced evolution strategies with covariance matrix adaptation that form a promising option if the controllers can be represented by real-valued vectors. In the future we will investigate the evolution of controllers using distributed evolution and the combination of the encapsulated and distributed variants (island model with migration).

In conclusion, we can state that the two greatest challenges embodied, on-line, on-board evolutionary systems have to face is the short time period for evolution and the very noisy fitness evaluations. Technically speaking, the robots can only

evaluate a few candidate solutions (genomes) in total and cannot perform enough re-evaluations. The source for both problems lies in the *physical* constraints of the system: a robot can only use one controller at a time and it should be using it for a "long" period to gain solid information about its quality. Thus, a possible cure for these two challenges is circumventing those physical constraints by incorporating and using a (possibly rough) simulator inside the robots for preliminary candidate assessment; only genomes that pass this quick test are further evaluated in real life. The costs will occur in the increased storage, memory and CPU power. The benefits will be the increased number of candidate solutions that can be evaluated and the increased number re-evaluations per candidate solution. It is very likely that both will contribute to more powerful evolutionary search still fitting within the limited physical time frame.

5.3 Artificial Sexuality and Reproduction of Robot Organisms

Christopher Schwarzer, Christoph Hösler, Nico K. Michiels

We envision an autonomous robot that is much more flexible than a smart machine; one that can adapt to unforeseen situations by reconfiguring itself. Our approach to this vision is the robotic organism: a robot entity that consists of many modular, autonomous robots analogous to the cells in a biological organism. This robot swarm can reconfigure itself both by changing controllers as well as its physical configuration by docking and as such can form a myriad of possible solutions. However, this vision imposes many new dimensions in design complexity. The most striking additional degrees of complexity are the robot interactions within the swarm, the physical configuration to create various body shapes and the coordination among many different robot cells within the robotic organism. In addition to the complexity of single robots with sensory processing, actuator coordination and artificial intelligence, the design of a robot organism is much more complex.

One method to obtain optimized solutions in the face of the drastic increase in design complexity is to employ evolution. Since the multi-robot organism is inspired by multi-cellular life, it makes sense to borrow inspiration from biology. Biological evolution can be seen as a design process that has developed an uncountable number of well-adapted organisms and, as a concept, can be used in robotics to take over parts of robot development. Instead of designing the robot in its entirety, we just design an elementary system with mechanisms that make it adapt itself to whatever challenge it is faced with. While there are other good approaches for this, such as learning, evolution is promising because natural evolution has developed fascinating complex multi-cellular organisms.

A key attribute of this evolutionary approach is to run it directly on the robot (on-board) while it is performing in its task (on-line). While this allows the robot swarm and organism to be truly independent and autonomous, it also requires that the evolutionary process is very fast and efficient as it is running in real-time. The faster the evolution, the sooner and better the robots can perform in their task. However, we are facing here a large difference to natural evolution because it has much

more resources available, both in time and number of individuals, than current robot swarms. Also, the pressures in natural evolution are much more numerous because many species cooperate and compete together for many types of resources and natural environments are less structured than man-made ones. Therefore, it is a major concern to employ an evolutionary process with a high evolutionary speed and efficiency while keeping some of the adaptability to complex environments. We can achieve this by using inspirations from biology in the form of bio-inspired evolutionary mechanisms.

Artificial sexuality is one such mechanism with promising potential and in this section, it is explained how sexuality can be adapted to robotics. Additionally, an entire set of bio-inspired evolutionary mechanisms is introduced here that is also related to artificial reproduction but can also be applied to asexual reproduction.

5.3.1 The Role of Sexuality for Robots

Sexuality in robotics is the system of exchanging genetic information between robots. While we might hold a romantic impression about sexuality, its essence is the *exchange of information*. The transmission of any genetic information between robots is already sexuality. Sexuality is a term from biology to describe analogous processes in artificial evolution. Sexuality is actually present in most models of artificial evolution in the cross-over or recombination operations. In a model of artificial evolution without sexuality, genetic information may not be exchanged between individuals and the only way to create new genomes is to mutate the genome of an individual. For robotics, sexuality becomes obvious when robots communicate genetic information to each other; genetic information "leaves" one physical robot and is received by another physical robot.

Artificial Sexuality. For creating artificial sexuality in robots, it is necessary to have a robotic genome, an encoding of variable parts of of the robot controller. And the basic element to realize sexuality is an exchange mechanism of this genome, which can be efficiently done by data communication between robots, and genetic recombination operators. However, the more interesting aspect of sexuality, which is the focus of our approach, is the mechanism used by robots to choose their partners for genome exchange; in other words the sexual strategy and mating behaviour. For this we can draw upon lots of knowledge in natural evolutionary sexuality. For example, the anisogamy of gametic cells in nature (Parker *et al.*, 1972) can be translated into two elementary robotic sexual functions: male function sends gametic genomes; female function receives gametic genomes, creates zygotic genomes and produces offspring by transmitting zygotes to other robots. In nature, many different strategies can be seen regarding male and female sexual functions. Two notable and widespread strategies are hermaphrodism, where an organism has both sexual functions, and gonochorism, where each organism has only one sexual function. Both strategies have advantages and disadvantages, which are the focus of the implementation described in Sect. 5.3.3.

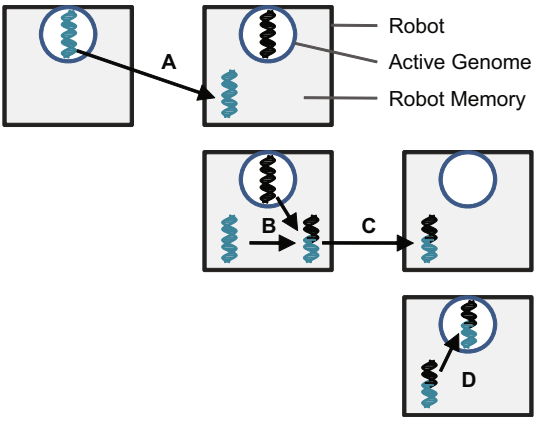


Fig. 5.25 Sex and Reproduction, the two modes of exchange of genetic information between robots. In sex, a gametic genome is transferred to another robot (A) and then recombined to create a zygotic genome (B). In reproduction, the zygotic genome is transferred to a different robot (C), where it is activated to take control of this robot and be evaluated (D).

Sex and Reproduction. The notion of artificial sexuality as genetic exchange overlaps with another concept of artificial evolution: reproduction. Reproduction is when a new individual is created with a genome constructed from the information of other genomes. In swarm robotics, this requires exchange of genetic information. Unlike in nature, offspring cannot be constructed from scratch and the number of robots in the swarm is limited and constant. Therefore, genomes have to be communicated between robots to redistribute them in the swarm. Think of the robot as a resource, like the shell of a hermit crab, that is required for an individual to live but when the individual dies, the robotic shell persists and can be used by another individual. Hence in order to reproduce, genomes are transferred between robots. When a genome in a robot is deactivated to reuse the robot for another genome, it is analogous to death. Likewise, the activation of a new genome to take over the control of a robot can be seen as the birth of an individual. Fig. 5.25 illustrates the differences between the genetic exchange of sex and reproduction in the robotic evolutionary context. In sex, the copy of an active genome is transferred to another robot where it is stored. The copied genome is then used to create a new genome, possibly recombining a number of genomes. In reproduction, a genome is transferred to another robot where it is activated. Also variations of the model are possible where the consequences of sex and reproduction are delayed when sperm and egg genomes are stored or passed on for later usage.
Any genome in this system falls into one of three categories:

Active — The genome is being used in a robot as part of the controller. The genome controls this robots behaviour and the performance of the genome is evaluated.

Gametic — The genome is a copy of an active genome. This genome can be a partial copy of a whole genome, analogue to haploid biological gametes. This genome is transmitted between individuals for being processed to create zygote genomes.

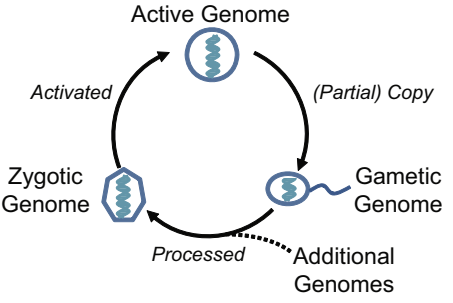


Fig. 5.26 Different states of genetic information. A complete genome is active and evaluated. Parts of an active genome are copied as gametic genomes. Genetic information from one or more gametes is processed to create a zygotic genome, which is a fully functioning set of genes which can be activated.

Zygotic The genome is a product of a processing operation which used other genomes as input. This a complete genome and is transmitted to a new individual where it can turn into an active genome.

Advantages of Robot Sexuality. The advantages of sex over asexual reproduction is a heavily debated topic in evolutionary biology. While it could not be proven conclusively that sex is better than asex, it is assumed that sex increases the speed of evolutionary adaptation (Colegrave, 2002). We think that robot sexuality, with implicit fitness by reproductive competition and bio-inspired mating mechanisms, is a strong addition or alternative to using classical approaches with an explicit fitness function, such as evolutionary algorithms.

Sex is ubiquitous in nature. It is very rare that species exclusively stick to asexual reproduction for long periods of time in evolution and examples of such species are known as the ancient asexual scandals (Judson & Normark, 1996). Virtually all organisms from vertebrates to invertebrates and even bacteria have some method to exchange genetic information. However, the reasons for the predominance of sex is a highly debated topic. The predominance of sex in nature suggests it is advantageous overall but theoretical models are not conclusively able to compensate for the disadvantages of sex (Kondrashov, 1993). A common theory is that sex increases the speed of evolutionary adaptation, especially when coevolving with other species. For example in host-parasite-interactions, sex allows the host to change the genetic footprint of its offspring to evade excessive adaptation and exploitation by the parasite.

From an engineering standpoint, the method of asexual reproduction seems more efficient because no effort needs to be expended for mating behaviour and genetic exchange mechanisms. However, as it can be seen in the field of genetic algorithms (Eiben & Smith, 2003), sex is a useful element in the form of crossover operators (Doerr *et al.*, 2008). Sexual recombination is known to speed up evolution because it is able to merge rare beneficial mutations that appeared in distinct individuals in the population. Beneficial phenotypes that emerge from a combination of mutations can be more rapidly produced and malicious mutations more easily purged from the population. In algorithmic terms, sex is utilized for coarse grained jumps in the search space to get into and out of local optima.

Lastly with sexual selection, sexuality offers many mechanisms to further increase the rate of adaptation. These mechanisms take place in the decision of the individual about who to mate and exchange genes with. Here, sexuality has the potential to predict good compatible genes using evolving heuristics; a mechanism can evolve to recognize honest and reliable indicators of partner fitness. A conceptual sketch about how this sexual selection can be applied to robotics is given in Sect. 5.3.4.

5.3.2 Artificial Reproduction

Reproduction is an essential element in natural evolution and the reproductive performance of natural organisms is the decisive factor in whether the species goes extinct or thrives and succeeds in evolution. Compared to evolutionary algorithms, artificial biological reproduction is just an agent-based view on evolution rather than a top-down, omniscient view. Thus any evolutionary algorithm does reproduction when genomes are copied as data structures and computing resources are allocated to evaluate them. A genome is reproduced when parts of it (after recombination) get additional resources for evaluation. Because swarm robotics is an agent-based approach, without a central supervisor, it is beneficial to adopt the view of reproduction for evolution.

Fig. 5.27 illustrates the fundamental pattern in agent-based and robotic evolution. Because the number of robots in the population is constant, the mean number of offspring is exactly one. Differences in number of offspring come from the implemented robot sexuality, which creates competition for the robots reproductive efficiency. The idea for the design of the evolutionary system is as follows: A robot which performs averagely in the current scenario should have one offspring. A robot which performs worse than most should have no offspring. A robot which has one of the best solutions currently should have more than one offspring, virtually by taking over offspring of less fit robots.

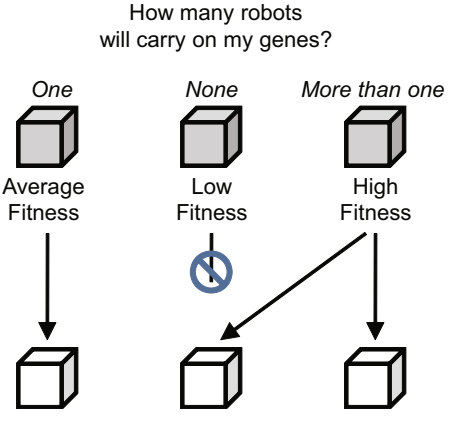


Fig. 5.27 The basic pattern of reproduction for a robot in embodied evolution. The average reproductive rate is exactly one offspring because the number of robots in the population is constant. A robot with high fitness must create more than one offspring, which implies that other robots have no offspring and thus low fitness.

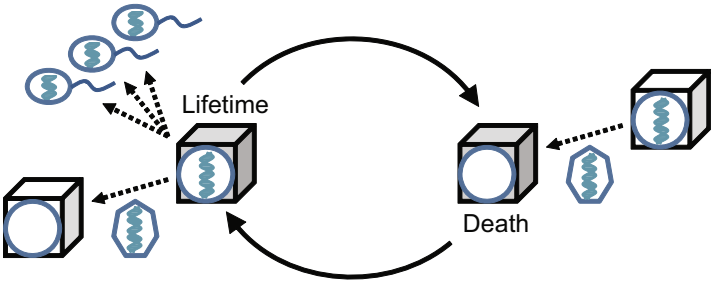


Fig. 5.28 The lifecycle of a robot. The lifetime of a genome lasts from activation in a robot until deactivation or "death". During lifetime, the genome participates in genetic exchange by creating and transmitting gametic and zygotic genomes. At the end of its lifetime, the robot goes into a temporary death state until a new zygote genome is provided, which is activated in the robot and takes over control.

Robot Life and Death. As indicated above, the creation of offspring is realized by a robot deactivating its current genome and instead activating a zygote genome, which controls the robots behaviour and is evaluated. Note that in any model of evolution, genomes have to be deactivated or their evaluation has to be stopped to free resources to evaluate other, potentially better genomes. This is an analogy to death and raises the question of how long a robot should live? The advantage of such a viewpoint can be shown in an example. Assume an evolving robot swarm where the genome in a robot is only replaced with another genome when two robots meet. Here an undesirable outcome could be that a particularly bad genome leads the robot away from other robots, potentially getting stuck somewhere. Without a limited lifetime, this robot is lost to the swarm and to the evolutionary process. Robot death can be realized as a default or fallback behaviour which helps to return the robot to the swarm. Note that it is also a bad solution to allow the genome to clone itself upon death if no zygote-genome is available, as it may prolong the active time of poor genomes.

With the adoption of a genome lifetime, it is possible to create a lifecycle of the genome, which is illustrated in Fig. 5.28. Genome lifetime begins with the activation of the genome in a robot. During life, the genome is part of the control of the robot and its performance is evaluated. This evaluation can be done explicitly, for example with a fitness function or by summing up a score over time, or implicitly by integrating mechanisms for reproductive competition, which will be shown in an example in Sect. 5.3.3. The end of the lifetime should be enforced by some mechanism and is marked by the deactivation of the active genome. The robot is then in a temporary "dead" state which means that this robot, as a resource of the evolutionary system, is now free to be used by a new genome. This notion of robot death may seem absurd but it is fundamental and it is subtly present in most approaches. For example in the original approach of embodied evolution by Watson (Watson *et al.*, 2002), genomes are overwritten by other genomes. This overwrite is the "death" of the overwritten genome because its original state is lost. It can also be seen in this example that the

dead state does not have to extend over a long period of time but it can be arbitrarily short. The goal is to keep death downtime minimal but the decision which genome to use to reactivate a dead robot is a big selection factor in the evolutionary system and should not be neglected.

5.3.3 Implementation of Artificial Sexuality on Real Robots

The concepts of a bio-inspired embodied evolution, which have been laid down in the previous sections, have been implemented on a swarm of up to 40 Jasmine III robots in the University of Stuttgart. These swarm robots have a cube-like shape with an edge length of 2.5cm. They are very quick with a two-wheel drive achieving a velocity of about 0.5 m/s and able to do a full turn in less than a second. They are equipped with a battery capable of providing over an hour of runtime, an ATmega168 microcontroller and 6 infrared sensors for obstacle detection and communication. With this platform, the feasibility of an evolutionary system which uses limited life time, robot death downtime, competitive reproduction and artificial sexuality could be shown and analysed.

Implementation. Our evolutionary approach has been embedded in a model of a reproductive cost-benefit trade-off, which is designed to reflect empirical observations about principles of sexual reproduction strategies. The primary aim of most living individuals is to spread their genes in the population. Factors like physiological constitution or environmental conditions build the framework of how to be successful at this costly task. Different species have evolved diverse methods to spend their energy in a competitive but efficient way. This is the playground of sexual competition and sexual conflict. In our scenario the competition happens in females and dead robots, as both genders accept multiple genomes for consideration but select only a good one for processing. The quality of the donor genome is equal to the amount of energy spent in its production. Therefore, a donor has to decide whether to distribute its fixed lifetime energy in few highly competitive genomes or invest less at more opportunities. This trade-off has been simulated in populations of gonochoristic and hermaphroditic reproducing robots at different densities. The idea was to test the effects of differences in the mating system and environmental perturbations on the evolutionary adaptation.

Physically, the genomic exchange is done via the robots' infrared communication modules. The genomic structure was kept very simple (one byte per gene) to enable a very fast drive-by communication mode and therefore a high mating frequency. The benefit of this is in keeping the generation time short and, thus, speeding up evolution. Besides the genome exchange, the robots task has been limited to explorative movement.

The populations were kept in plane walled areas of different sizes but similar rectangular shape. Each Strategy$\times$Density combination was simulated for at least 50 minutes. At a simulation start, the robots have been placed equally dispersed with the maximum Virtual Life Energy (VLE) of 100, 50/50 gender ratio and random genome. For each second in the alive state of the behaviour cycle, a robot loses 1

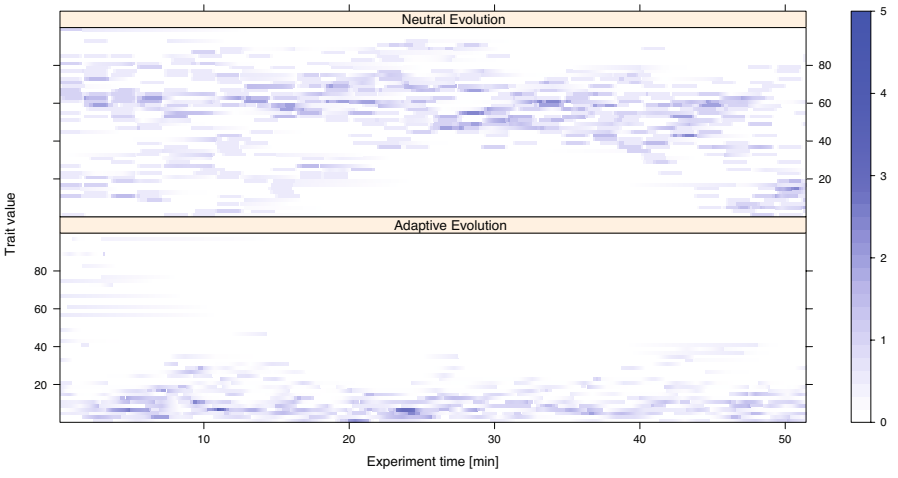


Fig. 5.29 The distribution of the trait MaleEffort in the population over time. In the upper plot cost and benefits of the trait are disabled and they are enabled in the lower plot. Without cost and benefits, the pattern of the upper plot shows a highly scattered, random distribution of the trait value, which indicates neutral evolution. With cost and benefits of the trait, the distribution in the lower plot is mostly restricted to the lower range, which shows an adaptation of the trait to low values.

VLE. The loss due to reproduction effort are encoded by a gene which is affected by recombination and mutation and therefore able to evolve inside the population. If a robots VLE drops below the minimum requirement for a reproductive task, it immediately switches to the death cycle in order to provide itself as a new offspring resource.

Observations. A first series of experiments with the described system showed that the populations adapt to the modelled competition (Fig. 5.29). Furthermore, both the density of the population and the sexual strategy in question have an influence on the trait expression value and the investments able to make per lifetime. With increasing density, the time to find a potential partner decreased for both strategies. Though the hermaphroditic system has to maintain two reproductive functions, the advantage that every conspecific is a potential partner, seemed not only to outweigh this drawback, but also to make this strategy superior in terms of a faster generation time. Most interesting was an unexpected observation in populations at high density. The temporary accumulation of robots at narrow spaces created barriers that were difficult to pass and made potential mates more difficult to find. Unsurprisingly, gonochorists suffered more from this environmental induced change than hermaphrodites.

Apart from the simulation results, preliminary tests also provided valuable insights in general properties of a bio-inspired robot population. First, as do natural populations, robot populations using the concept of life and death have a minimal viable population size. Too few robots lead to extinctions at a very high frequency.

Second, a reliable low range wireless communication network is a crucial property of an optimal robot swarm. Failures in the communication of reproductive mechanisms can be costly for individuals, which are situated in time and space. The speed of the entire evolutionary process depends on the speed and amount of genetic exchanges.

5.3.4 Evolutionary Engineering

The design and tuning of an embodied evolutionary process is an open challenge because there are many evolutionary mechanisms available which can be adapted from natural evolution but there is little known about the conditions and dependencies under which those mechanisms are beneficial in an artificially constructed evolutionary process. For example, while the concept and implementation of sexual genetic exchange are transferred conservatively from biology and common practice (transfer and recombination of genetic information), there are many open details not yet taken into account. What is the most suitable sexual system (hermaphroditism, gonochorism or something completely new)? Should selection work passively by costly reproduction or designed by the human observer? Another big question is the design of the sexual competition. How is the reproductive competition realized to give good genomes an advantage? Should competitive advantages be evolvable (including precopulatory gifts or postcopulatory harm ((Vahed, 1998), (Michiels & Newman, 1998))? And, as a third example in context of multi-robot organisms, how to design the evolutionary advantages of multicellularity?

It is an extensive task to design an efficient artificial evolutionary process for a given application scenario. This is comparable to the difficulty of developing a good fitness function that actually represents the goals of the designer. In embodied evolution the goal is rather implemented with the design of the evolutionary system and the competitive mechanisms therein. Thus, embodied evolution works without the need for an explicit function that evaluates the fitness of individuals; instead, it is the amount of offspring an individual manages to produce that describe its fitness. Rather than testing the individual actively with a fitness function, it is the active agent that struggles to prevail and reproduce well within passive selection mechanisms. The major goals of this "Evolutionary Engineering" are: high speed of evolutionary adaptation in a dynamic environment, high quality of evolved solutions, low loss of robot work time for reproductive tasks. Much more investigation is needed to evaluate different approaches and mechanisms and to gain experience in their usage and applicability.

Design of Embodied Evolution

In this section suggestions are given for the design of an evolutionary system of embodied evolution with robots. These are based on the discoveries from the work done in the aforementioned experiments on Jasmine III robots to include facets that proved useful and provided improvements for apparent flaws.

Challenge Gradient. A gradient of increasing challenges from one stable condition to the next is required by evolutionary adaptation to be efficient. Some evolutionary transitions might seem to have taken a steep step because of the severe differences in the required adaptations like those from fish to land animals. But in as in this example actual evolution is always expected to have taken place in a "gradient zone": vast areas of shallow waters, which were rich in nutrients because of falling leaves of trees but low on dissolved oxygen. This habitat could be entered by fishes and they could evolve adaptations which would give them gradual advantages like lungs to breath atmospheric oxygen, and simple limbs to hop from a dried out pond to the next (Clack, 2005). Similarly, the design of an evolutionary process for robots requires that the steep goal we set can be reached by a series of small steps. This is analogous to the problem of smooth fitness landscapes in evolutionary algorithms but, in embodied evolution and robotic organisms, we literally speak of smooth landscapes of the environment. It will be virtually impossible for a swarm of individual robots to evolve into a legged organism of dozens of robots that climbs over high walls without offering them a series of smaller walls first.

Limited Lifetime. In Sect. 5.3.3, we have shown a basic approach on regulating lifetime by basing it on a virtual life energy. Lifetime begins with an amount of life energy which depletes over time; there is a regular, steady base cost and additional costs depending on behaviour. Lifetime ends when the energy is depleted, however problems arise when means are available for gaining life energy, for example as rewards in a goal oriented approach. If gaining energy is possible, it is necessary to prevent infinite lifetimes. One possible solution is to use additionally a hard limit to the lifetime, like a maximum age.

Death. As stated before, genome death is essential and unavoidable in the evolutionary system. However it is a large hit to the efficiency of the system if robots spend much time in a dead state without a genome in control. The difficulty arises because there has to be a competition about which parent supplies a zygote genome for the dead robot, for example with a bidding contest. However such a process takes time if a sufficient number of competitors should be involved, especially if communication means are limited in the robot swarm and interested parties have to find the dead robot. A simple solution would be to give the dead robot a fixed behaviour to make it more likely to meet other robots. A more advanced solution would be to begin the competition for the robot before its lifetime is over. For example the robot would carry besides its active genome a slot for the successor genome and parallel to the lifetime of the current genome there is competition for the successor slot. However, any interference between the behaviour of the current genome and the competition for the successor may lead to complications. Ideally, the present genome behaviour and competition for successor genomes should not influence each other.

Development. Organisms of many natural species undergo a developmental phase at the beginning of their life cycle. While the main purpose of this development is to build the organism according to its building plan, the developmental phase is

also a first test of the functionality of this genetic setup. Natural fertilized egg cells often contain severe gene defects that prevent the formation of an embryo, or lead to its early death. In the same manner, evolutionary controllers could be forced to die early as well. The limited resources in the evolutionary process can be used more efficiently if the basic functionality of a genome and the corresponding controller can be tested cheaply early in the life cycle. If these tests fail, the genome can be classified as lethal, leading to an immediate death and freeing the robot as a resource.

Virtual Energy. The virtual energy is the currency of the evolutionary system. The basic design principle of passive selection is that reproduction costs energy and reproductive success is linked to the amount spent and the efficiency of the reproductive strategy. The success of a reproductive strategy and investment depends on those of other robots, a relative higher investment or better efficiency gives more offspring. So one part of virtual energy is spent on tasks directly related to reproduction, like mating and creating offspring. The other part is spent on tasks and activities that do not relate to reproduction directly, like base costs for staying alive, harvesting energy or movement. The idea is that the activities that are costly in reality also have a somehow related cost with this virtual energy. For example using a

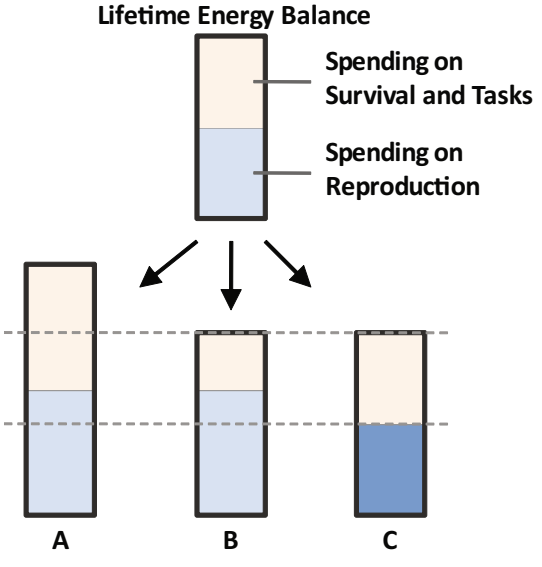


Fig. 5.30 The distribution of the virtual life energy investments of an individual can be balanced over its entire lifetime. A higher investment in reproduction relative to the competition yields more offspring. There are three strategies to increase reproductive success:
A: Gain more energy and thus have more total energy to spend.
B: Solve other tasks with less energy so a larger ratio of total energy can be invested into reproduction.
C: Improve the reproductive strategy to increase the efficiency and effective fitness with the same amount of energy investment. This is symbolized by a darker colour of the reproductive investment.

very power consuming motor or doing computationally expensive processing may be designed to incur energy costs and so evolution will optimize their usage. Because of costly reproduction, evolution always optimizes energy efficiency which is the reason for many highly efficient structures seen in nature that baffle engineers, for example shark skin (low friction in water), spider silk (high tensile strength) and the visual cortex of humans (vision processing).

There are three possible strategies for fitness optimization, which are also shown in Fig. 5.30. [A] Gain more energy, by harvesting more rewards from the environment for example, and thus more energy can spent on reproduction. [B] Spend less energy on other tasks by optimization so a higher portion of energy can be spent on reproduction. [C] Increase efficiency of reproductive investment by optimizing reproductive strategy. The first two ways [A] and [B] are related in that they both revolve around optimizing other tasks, either gaining more rewards or being more efficient. This can be easily translated into goal driven optimization, performing a desired and rewarding task either more often or faster for example. Strategy [C] differs slightly because it revolves around optimizing reproductive behaviour; it is more an optimization of the evolutionary process itself - a meta evolution. Here lies an advantage of artificial sexuality because it is self-optimizing to find better and more efficient selection criteria. For example, the evolutionary system we implemented on Jasmine III allows only strategy [C], individuals can evolve the tactic of their mating investment. Strategy [A] could be added integrating evolution of a foraging behaviour, to gain more energy than their base amount. Strategy [B] is available if the robots could also evolve their movement pattern, to spend less energy on finding mates.

Sexuality. Mating in sexually reproducing animal species often involves harsh sexual selection. Usually females only mate with the "best" males according to some sexual criterion, for example females of a species of poison frogs mate with males with the best quality of calling and it was found that the call quality is a reliable predictor of good offspring quality. The quality of frog calls improves with age, and only frogs that are able to escape predators and find food well grow old, thus call quality is an indicator of the ability to survive (Forsman & Hagman, 2006).

Also other examples of chosen phenotypes in nature show that those indicators must be costly or else it would be easy to cheat by developing the preferred attribute without any correlation to fitness. The classic example of a costly sexual trait is the long, ornate tails of peacocks, which costs a lot of energy to produce and is a serious hindrance of mobility. However, males with long tails show their strength and health by being able to maintain such a luxurious attribute and are thus preferred in mating.

Thus, traits that are used in mate selection must be costly, honest and either directly linked to fitness, like the poison frogs call quality, or indirectly, like the peacocks tail. These requirements can be transferred to embodied evolution as an evolvable criterion that determines mate preference. Robots can be built to display individual traits that are physically detectable with some perceptual modality but it is probably more simple to virtualise a display trait with communication. In some kind of mate protocol, the robots expose some information about their internal state,

for example with a neural network controller this could be the values of certain neurons and the number of neurons or links; some characteristic footprint of the network. Another robot receiving this message may then base its willingness to mate depending on the provided information. This mate preference can be implemented efficiently in a postcopulatory manner as shown in Fig. 5.31 by storing genomes of sexual partners in a gametic pool together with a rating of each partner. Now this preference itself is able to evolve. Assume that mate preference is random in the beginning and random values of this network footprint are preferred. However once an individual evolves to prefer a trait that correlates with good fitness, this individual has a strong advantage. This individual spends its energy for reproduction more efficiently because it prefers to create offspring with partners that indicated good fitness and thus the produced offspring has a higher fitness. This is what is expected by the theory of sexual selection, a mate preference evolves to choose those traits that are honest indications of good genes and this speeds up evolution.

Goal-Driven Sexual Evolution

How does sexuality help the robots solve a given goal? First of all, let us assume a scenario where we, as robot developers, do not know beforehand the best solution and the robots have to adapt autonomously to achieve their goal. Evolution is a mechanism for autonomous adaptation and sexuality is an addition to increase the rate of evolutionary adaptation. We propose to use a virtual energy currency in the robots to enable goal-driven evolution. Robots need this energy for reproduction and having more energy available for reproduction is a selective advantage. A robot is rewarded with energy for succeeding in certain activities that help in achieving the goal and thus has more energy available to spend on reproduction. Alternatively, costly and essential activities can be optimized to consume less energy like, for example, exploration or mating. Unlike an explicit fitness evaluation however, energy rewards do not directly translate into offspring. Consider an individual that spends its entire time working and harvesting rewards but does not participate in

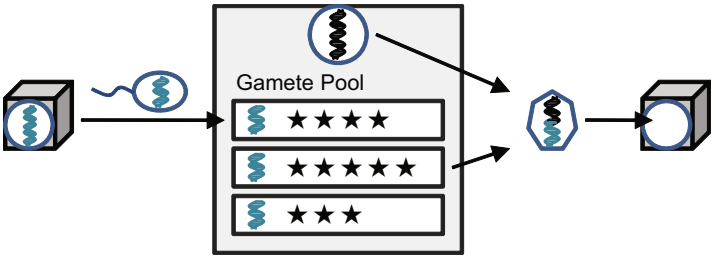


Fig. 5.31 Concept of postcopulatory mate choice via a gamete pool. During mating, the robot receives the gametic genome and also characteristic information of its partner. The robot rates this information and stores the gametic genome with the rating, symbolized here with stars. Higher rated genomes have a higher chance of being picked from the pool to create offspring.

reproductive competition—the accumulated rewards are lost when its lifetimes expires. Thus, the gained rewards are an incentive that has to be transformed into higher fitness by finding a balance between exploiting rewards and reproductive competition. The sexual mechanism itself evolves to increase the efficiency of reproductive investment, by making a smarter mate choice for example. This in turn increases the rate of adaptation even more as time is saved by focusing on reproducing with partners that promise offspring with the potential to gain a lot of virtual energy, in other words which are closer to reaching the goal.

5.3.5 Evolution of Multicellular Organisms

One interesting application of sexuality and bio-inspired reproduction is the evolutionary transition of single cells into multicellular organisms. Swarm robots are ideal for this because the robot swarm is like a population of single celled organisms that can exploit synergies by cooperation. While common swarm robots can only cooperate at the level of communication, by exchanging knowledge or coordinating movement, much tighter cooperation is available with reconfigurable robotics. Physical docking and linking between robots allows the exchange of energy and the formation of larger articulated structures, which is a level of coordination similar to multicellular life.

Advantages of Few-Celled Organisms. Natural evolution has produced the design innovation of multicellular organisms out of single-celled elements and we can attempt the same for robotics. The goal is to evolve robot controllers that produce small robot organisms of 3–4 robots that exploit some advantage. The challenge in this goal is to start with a plain robot swarm of independently, autonomous robots that are stable in this stage. From this single cellular starting point, evolution develops multicellularity because it has advantages in the evolutionary system. This setup is also shown in Fig. 5.32. Note that advantages of a robot organism with only

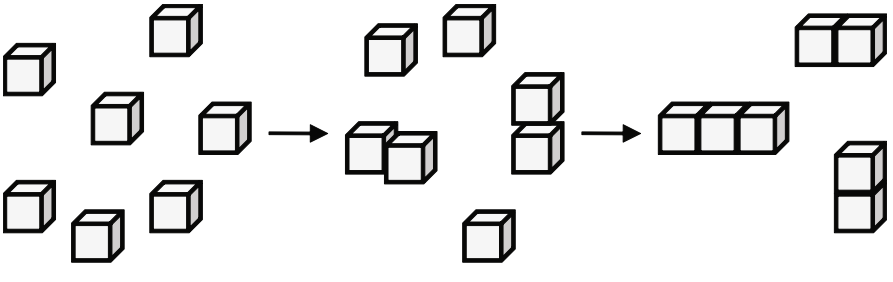


Fig. 5.32 Robotic evolution of multicellularity. Starting with a swarm of individual (single-celled) robots, evolution produces simple robot organisms consisting of a few robot cells. Incentive for docking can be implemented by reproduction; docking is needed to reproduce. In addition to genes that allow keeping the link after creating an offspring, reproductive advantages for multi-robot organisms must be integrated, like longer lifetime, to make this evolutionary transition possible.

few cells are less powerful than those of organisms with many cells. Having only a few cells limits the formation of articulated structures, like legs or snake-like bodies. But still synergies are possible depending on the hardware design. For example energy consumption can be reduced by turning off processors in linked robots and locomotion can be stronger because of joint drive power. The design of the evolutionary system can simply model these advantages for linked organisms with lower costs for certain actions and longer lifetimes.

Different Swarm and Organism Environments. One approach to this evolution of robotic multicellularity is now explained. Assume that there are two environmental conditions for the robot swarm: one swarm condition in which individual robots are more efficient and one organism condition in which linked robots are more efficient. Imagine for example solar powered robots that can conserve energy under low light conditions by linking and turning off some hardware. Such a population of robots is deployed in the swarm environment. They are equipped with an evolutionary controller which initially produces swarm behaviour and which is evolutionary stable in the swarm environment. Then the environmental conditions for organism mode is gradually introduced and the robot swarm should adapt by evolving multicellular shapes. To give a general incentive for docking with each other, reproduction and sexuality are very useful tools in the design of the evolutionary process. Reproduction can be designed to require a docking manoeuvre to transmit the zygotic genome. The docked transfer of a zygotic genome should be either the only way, or have a competitive advantage if other, wireless, ways of transmission are available. This serves to establish docking as a regular activity, which makes the transition to multicellular, permanently docked units much smoother. From the occasional, short-term docking, a mutation of the genome can be designed that gives a likelihood of keeping this docked state. Such a mutation will become stable in the population if there are advantages of the dual-cellular state, like this team of two robot cells has a longer lifetime than operating independently.

Organism Mode of the Controller. Given sufficient mechanisms of information exchange between the cells, like virtual hormones, the two robots can recognize being linked and controllers can evolve to operate such a minimal multicellular organism. This is the most critical point in the evolutionary transition because the controller of a single robot has complete autonomy but as soon as two robots are linked and want to perform as one entity, there are two controllers that are dependent on each other. It is easiest to assume that the controllers assume certain roles, that one controller takes the lead and the other submits. Though it is conceivable that other distributions of roles are possible including roles that are limited in time or to specific functions. However with the concept of reproduction by docking, certain plain approaches are an interesting option. For example, a parent robot creates offspring in another robot by docking. The linked state is kept and the parent robot is now leading the child robot for a while until the child matures and separates. This behaviour would match theories about parental investment and parental care in biology.

5.3.6 Sex and Reproduction of Symbiotic Robots

For sex and reproduction of a multi-robot organism, we have to ask ourselves about the main focus of this multi-robot entity because our vision of symbiotic robots can be split in two parts with very different reproductive implications (Fig. 5.33). One part of the vision is that robots form a multicellular organism with high cooperation and dependence between cells. This organism should have cellular specialization in the way that, for example, only some cells use certain hardware to conserve energy, like some cells do computationally expensive sensor processing and planning (brain), some do actuation (muscles) and others might just serve as skeletal elements and energy storage (bone and fat). And there is another part of our vision, that the robot modules form an aggregated structure just adapting to the current situation, may it be a distributed swarm for exploration, a bridge for crossing a gap or a closely packed block for hibernation. A natural example of organisms that can perform such a structural cooperation are social insects, like ants building bridges with their own bodies or the slime mold *Dictyostelium discoideum* whose amoeboid individuals can aggregate into a multicellular slug for migration.

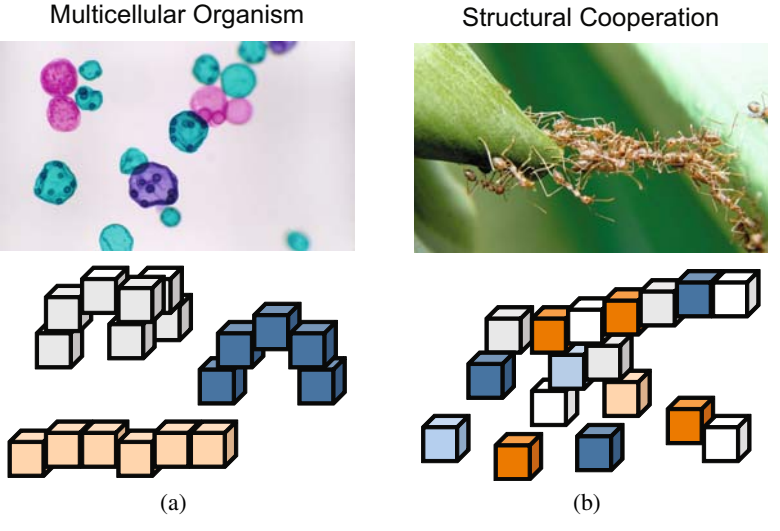


Fig. 5.33 Two possible approaches of symbiotic robotics: **(a)** A complex multicellular organisms with and genetically identical but highly specialized cells. Such an organism keeps its shape during its lifetime. The natural example is any multicellular organism like humans for example. **(b)** A robot swarm that aggregates to solve environmental challenges by using structural cooperation. All robots of the population are individuals with their distinct genome. The robots aggregates into an ad-hoc shape that is formed and maintained just for a specific situation. As biological example might serve the society of weaver ants (genus *Oecophylla*), which are able to form a bridge with their bodies.

However, there is no known example in nature of a species that has both abilities, tight symbiosis and ad-hoc aggregation and disaggregation of the same units. Note that all examples of organisms that dynamically aggregate into structures do not actually form a multicellular organism, these are rather swarm-like coordinated movements and specialization between participating individuals is low. Also all cells that separate from a multicellular organism always serve reproduction and form a new organism by fusion (sex) and cell division but never by aggregation. The lack of natural examples may be seen as an indication that it is more difficult to evolve organisms that can be both a swarm of independent units and flexibly aggregate into an organism of specialized cells. A theory that might explain this is Hamilton's Formula (Hamilton, 1964) which predicts that the evolution of altruism is relative to the relatedness of participating parties. In other words, the more genes organisms have in common, the higher is their potential for altruistic cooperation, which is required for tight cooperation of cells in an organism. This theory can explain the various forms of cooperative systems in nature among organism, in the order of increasing cooperation and relatedness: Animal swarms are all conspecifics, colonies of social insects are all siblings and cells of a natural organism are all clones.

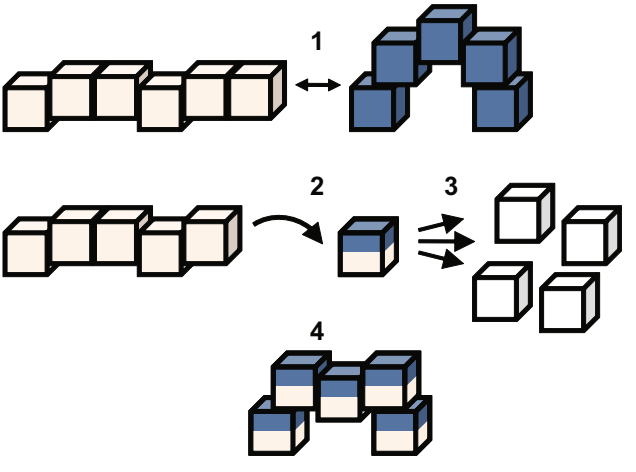


Fig. 5.34 Example of sexuality and reproduction of a multi-robot organism with a homogeneous genome.
1: Two organisms mate and exchange genomes. One or more robot cells of the organism become egg cells with a new genome.
2: The egg cells are detached from the organism and become independent.
3: The detached egg searches for lone robot cells. These come from dead organisms which have fallen apart into separate robot cells.
4: The found cells are assimilated by overwriting them with the genome of the egg cell and are used to build a new multi-robot organism. This ontogeny is initiated by the egg cell and can form a new organism with a different structure.

Multicellular Organisms

We assume that the way to achieve highly complex cooperation between and specialization of the robots is in the form of a multicellular organism in analogy to a proliferated germ cell. With a copy of a single genome in each cell, negotiations and stability have a higher chance to get reached. Reproduction in such a structure is quite simple with a homogeneous genome forming the super-organism. The evaluation of the organisms performance is directly related to this genome and any reproductive action uses this one genome.

Robotic Ontogeny. An example of the envisioned reproduction of such an organism is shown in Fig. 5.34. Two organisms mate and exchange genomes. Some cells of the organism can be considered egg cells, recombining the received genome and activating it; they are now fertilized egg cells. The egg cells are detached from the organism and begin their lifetime as an independent, new organism. First the newborn needs to undergo ontogeny and grow into a new organism. Lacking the capability for cell division, the newborn must assimilate other robotic modules. One possibility would be that when the lifetime of a robot organism is up, it falls apart into separated, "dead" robot modules that can be taken over by the newborn by linking with them and copying over its genome. The parents that lost some cells of their body by detaching the eggs can look for replacements in a similar fashion. The advantage of this process is that the detached egg has a relatively simple ontogeny by linking to the assimilated modules in a determined way to grow into the new adult shape. The alternative of taking over a parent organism would possibly require the reconfiguration of its cellular structure, which is more complicated. In this case it might be easier to disintegrate the parent and then starting growth from a single cell that reacquires the now separated modules.

The approach mimicking biological organisms also has limitations. A natural organism does not spontaneously disassemble because formation and disintegration of the organism is bound to the lifetime of this individual. The shape of the organism is also less flexible as it is determined by the genome. If such an organism encounters a situation that it cannot solve with its current shape, the whole organism first has to die for the modules to be reused with a different genome to produce a different shape.

Structurally Cooperative Swarms

The second approach is to focus on the single robots as autonomous individuals and their physical ability of structural cooperation. The idea is to extend the evolution of the distribution of labour in the population to the distribution of morphological functions. A biological inspiration is for example, the weaver ants which use their bodies to form a bridge (Fig. 5.33(b)). Despite this being a simple structure, it still has two distinct regions: a border where ants anchor the bridge to the leaf, and the middle section where ants link with other ants. Compared to a multicellular organism however, this aggregated structure is much more simple but it can be formed

and disassembled quickly, within the lifetime of the participating individuals. This ability for ad-hoc aggregation makes this approach very interesting but the open question is how complex those cooperative structures can be. There are fascinating examples of swarm cooperation in nature but the overall structural complexity rarely exceeds that of a huddle of individuals.

From a genetic point of view, the cooperative structure would consist of heterogeneous individuals, each with a different genome which has important complications considering reproduction. Such cooperative behaviour will not be sustained in evolution if the reproductive costs and benefits for participating do not match up. A robot that cooperates with others, to form such a bridge for example, has costs for this cooperation because it could evolve to stop cooperating and use the time instead for harvesting and reproduction on its own. What are the reproductive advantages of participating in structural cooperation for the individual? The entire aggregated group might gain a reward but as soon as it comes to reproduction the individuals are competing harshly against each other and selfish behaviour will likely appear.

Social Cooperation. The dilemma of selfishness can be tackled by integrating social rules to provide enough incentive for the individuals to cooperate and enough reassurance that their investments will not be abused by others; similar to a human company where hundreds of employees cooperate with large investments of each individuals lifetime because of social rules. Note that the prominent examples from social insects alleviate the issue by only allowing the queen of a swarm to reproduce. The members of one swarm are actually all siblings and their reproductive gain is assured because they share half their genes with the queen. Thus for a swarm of truly heterogeneous individuals, social ruling has to be more advanced than that of social insects.

Cooperation and Competition Phases. A possible approach to solve the conflict between our desired structural cooperation and reproductive competition necessary for evolution can be to establish two distinct phases by social rules in the swarm. The swarm undergoes structural cooperation where rewards are gained and distributed among the participants but then also a phase of competition can be allowed. The individuals of the aggregated structure fall apart and compete for reproduction where individuals that were especially helpful during the cooperative phase have a competitive advantage by having earned higher rewards for example. After competition, mating and reproduction has taken place the swarm unites again in the cooperative phase to replenish their energy.

Overall, the structurally cooperation of a swarm is the more powerful approach because it is more flexible than a multicellular organism. In theory the ideal is conceivable: a swarm that can transform into any complex, articulated aggregated structure at will. However, there are many open questions about how to achieve this. Currently it seems plausible that for the given effort there is a trade-off between organism complexity and aggregation flexibility.

5.3.7 Conclusion

In this section we discussed why sex has its definite place in the evolutionary process, both natural and artificial. In natural evolution, sex increases the rate of adaptation and helps to deal with unforeseen challenges. Thus, artificial sexuality has a high potential to improve the evolution of robots in their behaviour and their structural cooperation. Sexuality is a starting point which allows us to adapt many evolutionary mechanisms from nature into robotics which promise further increases in the speed of evolution, like gender expression and mate choice. Yet, it is a challenging task to design the rewards and costs mechanisms for such a bio-inspired reproduction scheme, similar to the difficulty of designing a fitness function in traditional evolutionary algorithms. Although, the ability of sexual reproduction to self-optimise mating strategies may overcome this burden. We are just beginning to understand how many of these evolutionary mechanisms can be employed efficiently in robotics and have yet to discover their implications and opportunities, which also benefits research in evolutionary biology.

Regarding multicellular robotics, we see two approaches: A multicellular organism like natural organisms which can be complex but stay in one shape over their lifetime; and a structural cooperative swarm like an ant swarm that forms simpler shapes but can aggregate and disintegrate at random. We assume that investigations in both of these directions are necessary as an intermediate step before a hybrid approach can be reached that combines both advantages: complex organism and spontaneous change of shape.

From the current perspective, sexuality seems costly considering the limitations of current robotics, especially short runtime and small population, but future advances in technology will reduce those limitations. And the benefits of sexuality will become stronger once the robotic application scenario leaves static environments and the robots face unknown, dynamic challenges.

5.4 Self-learning Behavior of Virus-Like Artificial Organisms

Vladimir Red'ko, Serge Kernbach

Artificial organisms are represented by their genotypes and phenotypes. Genotype is basically a symbolic string, which codes structural, homeostatic, regulative and other functionalities. However, an organism consists of many aggregated robots with different genomes, the genotype of an organism can be considered as a cloud of these basic genotypes. Such a representation is very close to the ideas of Eigen's quasispecies (Eigen, 1971). Quasispecies are a genetic model of an organism, represented as informational strings, analogous to the RNA molecules. The quasispecies model can be considered as a simple canonical model of evolution with well defined scheme, which supposedly took place on the prebiological stages of evolution (Eigen & Schuster, 1979). This model is equivalent to the genetic algorithm (Holland, 1992), (Goldberg, 1989) without crossovers. The quasispecies approach allows us to understand asexual properties of artificial organisms and a

virus-like behavior of modules within an organism with a high mutation rate, see e.g. (Hoffmann, 1994).

More generally, considering artificial organisms as multi-genome virus-like systems enables us to concentrate on cooperative and adaptive genetic properties of simplistic co-working systems. In particular, this approach can be used for studying self-learning properties and represents the second topic of this section. Here aggregated and disaggregated robots are viewed as general swarm agents, which possess different controllers: adaptive critic design ANN in Sect. 5.4.2, ANN with similarity genes in Sect. 5.4.3, logical behavioral rules in Sect. 5.4.4. All these models implement simple behavioral primitives, characteristic for viruses: looking for energy, self-reproduction, local environmental prediction, competition or simple forms of self-protection (Carter & Saunders, 1997).

The first model is intended to analyze evolutionary properties of autonomous adaptive systems (Red'ko *et al.*, 2005b) and an interaction between learning and evolution. Agents are involved into a homeostatic regulation of an organism and are controlled by ANN based on the adaptive critic design (ACD) (Prokhorov & Wunsch, 1997). The ACD includes two artificial neural networks: model and critic. The model-ANN predicts the state of an environment for the next step, and the critic-ANN is used to select actions on the basis of model predictions. These ANNs can be optimized by both learning and evolution. Learning uses the reinforcement approach (Sutton & Barto, 1998), whereas evolution includes mutations of ANN synaptic weights and selection of best agents. It turned out that a non-trivial Baldwin effect (Baldwin, 1896), (Turney *et al.*, 1996) can be observed in the model, i.e. originally acquired adaptive policy of best agents becomes inherited over the course of evolution. In particular, some features, which are obtained by learning in initial generations, become inherited in the fifth generation. This means that ANN synaptic weights, which are initially adjusted by reinforcement learning, are then rediscovered during mutations and selections.

As an extension of this model, we outline evolutionary emergence of cooperation between agents (Burtsev & Turchin, 2006) during an aggregation phase of artificial organisms. Agent is controlled by ANN, which connects receptors to effectors. Each effector corresponds to a particular action. The agent spends its energy by performing any action and its the energy level is increased by a foraging. The genome of the agent codes the structure of neural network and synaptic weights of neurons. An evolutionary process modifies this structure and synaptic weights by mutations. The genome includes so-called "gene of similarity". This similarity gene is obtained by the agent from its parent; so the difference in these genes between the child and its parent is small because it is created only by small mutations. Families of agents with almost identical similarity genes are evolutionary formed and we observe an emergence of cooperation between agents of one family.

An interesting phenomenon was observed in the early version of this model, which does not consider similarity genes (Burtsev, 2002). Some evolutionary processes create surprising peaks in the size of population. Analysis demonstrated that these peaks arise because of competitive "fight" effectors disappear from control systems of all agents in the population. So, agents do not compete with each others,

instead they use their energy for other useful actions. However, duration of such peaks is small: fighting effectors are restored by mutations and the population size is decreased again. In the last model, we use the same structure of agents, however the agent is controlled by logical "if – then" rules (Red'ko & Beskhlebnova, 2009), optimized by reinforcement learning and through evolution in the population. As turned out, such agents can produce chains of rules with some logical inferences, leading to adaptive collective behavior and can be used during pre-aggregative and aggregative phases of artificial organisms.

5.4.1 Effectiveness of Evolutionary Optimization for Genetic Cloud

As already mentioned in the introduction, artificial organisms represent an aggregation of swarm robots. Each of these robots possesses its own genome. During aggregation into an organism, there are different scenarios of how these individual genomes can compose one organism's genome. Such a common genome is necessary for an effective macroscopic control, for example for macroscopic locomotion (Kernbach *et al.*, 2008b). We briefly sketch different possible approaches for a genetic transition between swarm and organism in Fig. 5.35.

In the first, most evident case, shown in Fig. 5.35(a), we assume that all swarm robots have the same the structure of a genome. Such an artificial genome can have either a simple form of "switches", which turn on or off some functionality, or have a complex structure of context-sensitive grammars. During aggregation, robots basically reconfigure this common structure. When leaving the organism, a robot can store common genetic memory or, alternatively, reset it by returning to its previous genome.

Another approach is shown in Fig. 5.35(b). Common genome of an organism is a combination of different genes from the swarm mode. Principles of such a combination are different (several possible approaches are considered in Sects. 4.3 and 5.3). In general case we can assume that all genomes possess a common part, programmed initially into all robots, and a variable part, which can be individually changed (only one of these parts is also possible). The last part creates a genetic variability in the population of aggregated or disaggregated robots.

In the last case, shown in Fig. 5.35(c), such an organism's genotype is in fact a cloud of different genetic codes from a swarm-mode. From the biological perspective, such an organism corresponds to a "colony" (Camazine *et al.*, 2003). In the genetic cloud, efficient individuals are less relevant and their collective effect is of interest. Such a formulation is very similar to the quasispecies model proposed by Manfred Eigen (Eigen, 1971) and developed further by (Eigen & Schuster, 1979). In this terminology, genome is an informational string, which codes different regulative processes, and we can use evolutionary algorithms to optimize them. Since quasispecies model is equivalent to the genetic algorithm without crossovers, this approach uses asexual way of reproduction and is similar to the behavior of viruses, see e.g. (Domingo & Holland, 1997). From this point of view, multi-genome artificial organisms may be considered as virus-like structures, or colony of viruses,

whose cooperative behavior is determined by a collective effect of the genetic cloud. Further in this section, we intend to demonstrate that such a genetic cloud can represent an efficient evolutionary and regulative mechanism for artificial organisms.

Firstly we estimate evolutionary rate and effectiveness of evolutionary search (Red'ko & Tsoy, 2005), (Tsoy & Red'ko, 2006a), (Tsoy & Red'ko, 2006a) by the quasispecies model, assuming that the symbols of informational strings take only two values: +1 or -1. The main assumptions of the model are as follows:

1. The evolution of a population of organisms $\{\mathbf{S}_k\}$ is considered, where each organism $\mathbf{S}_k$ is determined by a string of symbols S_{ki} , and the symbols take two values, $S_{ki} = +1 \; or \; -1$; $i = 1,\ldots, N$; $k = 1,\ldots, n$; N is the length of the strings; and n is the population size.
2. It is assumed that the fitness function $f(\mathbf{S})$ is unimodal and there is the optimal string $\mathbf{S}_m$, that has a maximum fitness value and fitness of any other string $\mathbf{S}$ exponentially decreases as the Hamming distance $\rho(\mathbf{S},\mathbf{S}_m)$ between $\mathbf{S}$ and $\mathbf{S}_m$ (the number of non-coinciding symbols at the respective positions of these strings) increases. The fitness function is defined as follows:

$$f(\mathbf{S}) = \exp[-\beta\rho(\mathbf{S},\mathbf{S}_m)], \tag{5.2}$$

   where β is the parameter of selection intensity.
3. The evolution process consists of a number of generations. In each generation, selection of the organisms into next generation (in accordance with their fitness) and mutations (random replacements of symbols S_{ki}) take place.

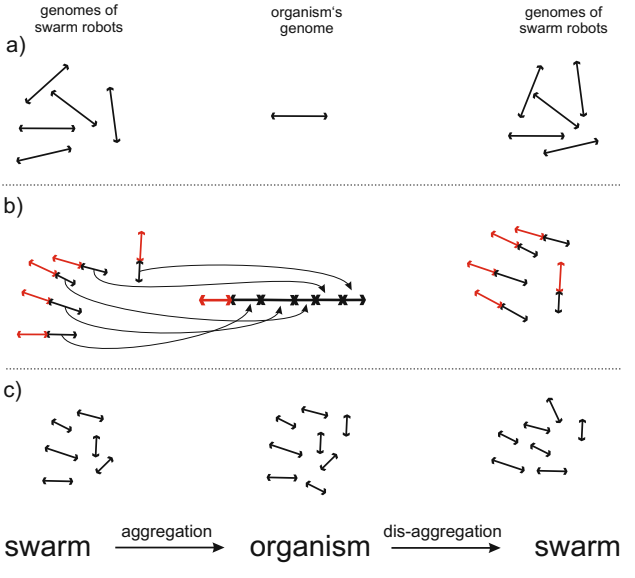


Fig. 5.35 Different approaches for transition between genome in swarm- and in organism-modes. **(a)** Switching genome; **(b)** Combining genome; **(c)** Genetic cloud.

4. The string length N and the population size n in a certain evolution process are invariable and large: $N, n \gg 1$.

The formal scheme of the evolution process in the considered model is represented in Table 5.4.

Table 5.4 The scheme of evolution

Step	Action
Step 0	The formation of the initial random population $\{\mathbf{S}_k(0)\}$. For each k = 1, 2,…,n and each i = 1, 2,…,N, a symbol S_{ki} is chosen at random; it equals either +1 or -1.
Step 1	Selection.
Substep 1.1	Fitness calculation. For the population $\{\mathbf{S}_k(t)\}$ (t is the number of the generation), $f(\mathbf{S}_k)$ is evaluated for each k = 1, 2,…,n.
Substep 1.2	Formation of the new population $\{\mathbf{S}_k(t+1)\}$. n strings are selected from $\{\mathbf{S}_k(t)\}$ with probabilities proportional to $f(\mathbf{S}_k)$ to form $\{\mathbf{S}_k(t+1)\}$.
Step 2	Mutations. For each k = 1, 2,…,n and for each i = 1,2,…,N, the sign of the symbol $S_{ki}(t+1)$ is changed with probability P; P is the mutation intensity.
Step 3	Organization of the sequence of generations. Steps 1, 2 are repeated for t = 1, 2,…

According to substep 1.2, a new population is formed as follows. Imagine that we have a roulette with the arrow. For each generation, the roulette is divided into n sectors so that the fraction of the k^{th} sector (in the entire disk) is $q_k = \frac{f_k}{\sum_{j=1}^{n} f_j}$, where $f_k = f(\mathbf{S}_k)$. Then, we start the roulette n times; every time, the number of the sector to which the arrow comes to rest is determined, and the string corresponding to this number is included in the next generation of the population. Thus, precisely n strings are included in the next generation. For each run of the roulette, the probability that the k^{th} string is selected for the next generation is proportional to its fitness $f(\mathbf{S}_k)$. Some strings can be selected several times; this means that the new population includes several descendants of these strings. It should be noted that the described model is characterized in full extent by the following four parameters: N, n, β, P.

Qualitative Picture of Evolution

Computer simulations demonstrate that if the mutation intensity is sufficiently small ($\beta \geq PN$, $1 \geq PN$), then the evolution can be characterizes as follows (Red'ko, 1986; Red'ko, 1990; Red'ko & Tsoy, 2005):

- initial distribution $Pr_0(\rho)$ with respect to ρ in the population (for $t = 0$) is close to the normal distribution with mean $<\rho> = N/2$ and variance $N/4$ (Red'ko, 1986); $<\rho>$ is the average (over the population) Hamming distance to the optimal string $\mathbf{S}_m$;

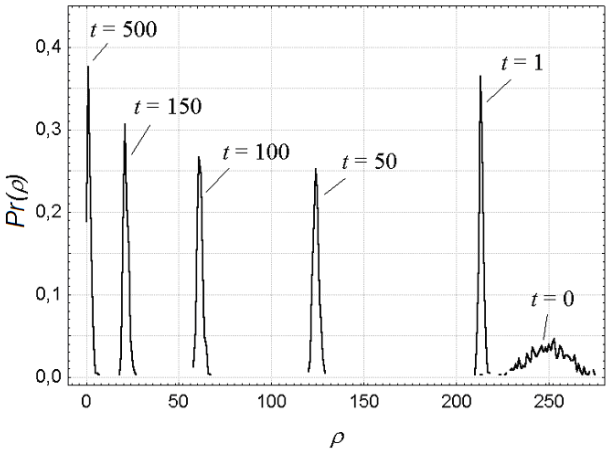


Fig. 5.36 The evolution of distribution $Pr(\rho)$ of Hamming distance ρ from the optimal string in the population. $Pr(\rho)$ is the fraction of organisms with certain value ρ in the population, t is the number of the generation. $N = 500$, $n = N = 500$, $\beta = 1$, $P = 1/N = 0.002$. The distribution for $t = 500$ corresponds to quasispecies.

- the dynamics of the distribution $Pr(\rho)$ can be characterized by two stages, rapid and slow;
- during the first (rapid) stage, organisms from the left side of the initial distribution $Pr(\rho)$ are selected and the distribution $Pr(\rho)$ contracts;
- during the second (slow) stage, new organisms with smaller values of ρ can appear in the population only through mutations, and the distribution $Pr(\rho)$ drifts to small values of ρ with low rate;
- the final distribution characterizes the quasispecies, that is the distribution in a neighborhood of the optimal string $\mathbf{S}_m$;
- at small selection and mutation intensities ($1 \gg \beta \geq PN$), the quasispecies distribution $Pr(\rho)$ is close to the Poisson distribution with mean PN/β (Red'ko, 1986).

An example calculation is represented in Fig. 5.36. We suppose that $2^N \gg n$, i.e., the length N of each string $\boldsymbol{S}$ is sufficiently large. The number of strings corresponding to certain species in the population is not large, and many species are absent at all. For this reason, fluctuations of the number of species are essential, and the evolution processes under consideration have stochastic character. In particular, the neutral selection, i.e., the selection independent of fitness, must be taken into account. Characteristic number of generations for the neutral selection T_n is of the order of population size n (Kimura, 1983; Tsoy & Red'ko, 2006a).

Analytical Estimations

Let us estimate the efficiency of the algorithm described above under the assumption that the population size n is sufficiently large, i.e.:

$$T_n \geq T, \quad \left[1-(1-P)^N\right]^n \ll 1, \tag{5.3}$$

where T_n is the characteristic time of neutral selection ($T_n \sim n$), T is the characteristic time of the convergence of the entire evolution process. The first inequality in (5.3) means that the influence of the neutral selection is sufficiently small. The second inequality corresponds to ignorance of the mutation losses of already found "good organisms" in the population.

Let us estimate characteristic convergence time T of the evolution process. For large N the value of T is determined by the second (slow) stage of the evolution (see Fig. 5.36). At this stage, new strings with smaller values of ρ arise because of the proper mutations and are fixed in the population through selection. Let us estimate the characteristic time t_{-1} during which $<\rho>$ decreases by 1. This time can be approximated by the expression

$$t_{-1} \sim t_m + t_s, \tag{5.4}$$

where $t_m \sim (NP)^{-1}$is the characteristic time during which the strings mutate and $t_s \sim \beta^{-1}$ is the characteristic time during which the strings with $\rho = (<\rho> -1)$ replace in the population the strings with $\rho =<\rho>$ in the course of the selection. The total decrease of $<\rho>$ at the evolutionary process is approximately equal to $N/2$. Setting $T \sim t_{-1}N$, we obtain

$$T \sim P^{-1} + N\beta^{-1}. \tag{5.5}$$

The total number of strings involved in the evolution is $n_{total} = nT$. Let us estimate the value of n_{total} for given N by choosing the remaining parameters β, P, n so as to minimize n_{total}. We assume that the intensity of the selection is sufficiently large, i.e., $\beta \geq PN$; this allows us to ignore the second term in (5.5). For mutation intensity we set $P \sim N^{-1}$; this corresponds approximately to one mutation per string in each generation. For this value of P, on the one hand, new strings appear in the population via mutations sufficiently quickly and, on the other hand, it is possible to ignore the mutation losses (the second inequality in (5.3) holds). Thus, we have $T \sim N$. We assume also that the first inequality from (5.3) is valid at the utmost limit, i.e., $T_n \sim T(T_n \sim n)$; in other words, we assume that the population has minimum admissible size for which the loss of felicitous strings caused by neutral selection is inessential. So, we have: $n \sim T_n \sim T \sim N$. With respect to all the assumptions we have:

$$T \sim N, \qquad n_{total} \sim N^2. \tag{5.6}$$

Several Results

In order to verify the analytical estimations (5.6), the dependence of the evolutionary search effectiveness on the string size N is investigated by computer simulations (Red'ko & Tsoy, 2005; Tsoy & Red'ko, 2006a; Tsoy & Red'ko, 2006a). The other parameters of evolutionary processes are set to values corresponding to the

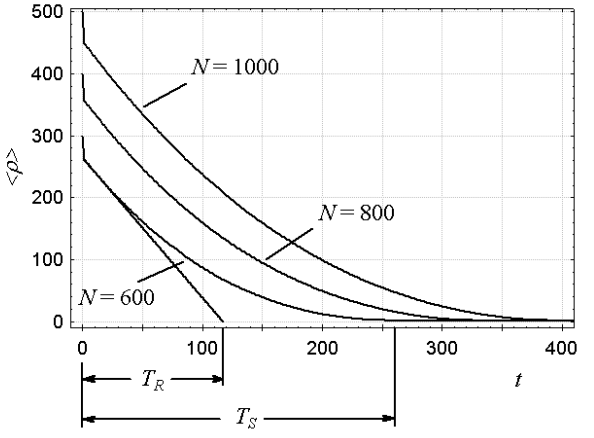


Fig. 5.37 The dependencies $<\rho>(t)$ for various values of N. The scheme for estimating the values T_R and T_S for $N = 600$ is shown. The dependencies are averaged over 50 independent runs.

conditions for obtaining the analytical estimates, namely: $n = N, P = N^{-1}, \beta = 1$. The computations are organized as follows. The time dependencies $<\rho>(t)$ of the average (over the population) distance to the optimum are obtained for various N (see Fig. 5.37), and these dependencies are used to estimate the characteristic time T of the convergence of the evolution in two ways:

1. The characteristic relaxation time T_R in the dependencies $<\rho>(t)$ is calculated from the initial slopes of these dependencies;
2. The time T_S of reaching the stationary value $<\rho>(t)$ attained at large t is evaluated.

The scheme of estimation of values T_R and T_S is shown in Fig. 5.37. In addition, the time T_O of the first appearance of the optimal string $\mathbf{S}_m$ in the population is determined. The resulting dependencies $T_R(N)$, $T_S(N)$ and $T_O(N)$ are shown in Fig. 5.38. We see that, for sufficiently large N, these three dependencies are approximately linear, namely, $T_R(N) = k_R N + T_{R0}$, $T_S(N) = k_S N + T_{S0}$, $T_O(N) = k_O N + T_{O0}$, where k_R = 0.1772, k_S = 0.3903, k_O = 0.3685, T_{R0} = 8.2709, T_{S0} = 38.7356, and T_{O0} = 2.1288. These dependencies are in a good agreement with estimates (5.6).

Comparison of the Evolutionary Search with Other Methods

Let us compare the evolution method of optimization of the fitness function (5.2) under consideration with the two simplest methods, sequential search and random search. The sequential search is organized as follows. We start from an arbitrary string $\mathbf{S}$ which symbols are S_i = 1 or -1. Then, for each i (i = 1, 2, …, N), we sequentially change the sign of the i^{th} symbol ($S_i \to -S_i$). If the fitness $f(\mathbf{S})$ at this

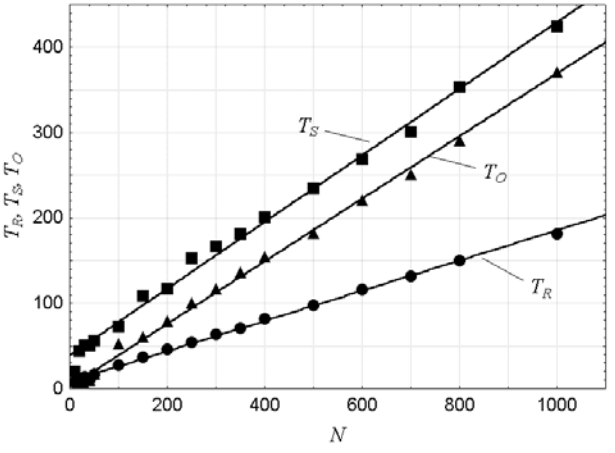


Fig. 5.38 The dependencies of relaxation time T_R , stabilization time T_S and time T_O required to find an optimal solution on the string length N. The dependencies are averaged over 50 independent runs.

Table 5.5 Comparison of the search methods efficiency.

Search method	n_{total}	n_{total} for N=1000
Sequential	N	1000
Evolutionary	$\sim N^2$	$\sim 10^6$
Random	$\sim 2^N$	$\sim 10^{300}$

change increases, then we accept the new value of the symbol; otherwise, we return the old value S_i. As a result, after N tests, we obtain an optimal string $\mathbf{S}_m$. Thus, for the sequential search, the total number of strings, which should be processed before the optimal string $\mathbf{S}_m$ is equal to N: $n_{total} = N$.

To find an optimal string by random search, the number of strings to be tested is of the order of 2^N: $n_{total} \sim 2^N$. The estimates obtained are given in Table 5.5. These estimates demonstrate that the evolution process as an optimization algorithm is "suboptimal": it does not ensure the maximal speed of search (for particular problems, more efficient algorithms are possible; in the case under consideration, such an algorithm is sequential search); nevertheless, it is much more efficient than random search.

Note that, although the estimates were obtained for unimodal fitness function (5.2), similar estimates can be made for the spin-glass evolution model, in which the number of local maxima of the fitness function exponentially increases with the string length N (Red'ko, 1990).

Thus, estimations of efficiency of evolutionary search in simple canonical quasispecies model, which have a virus-like nature, have been made. According to these

estimations, the optimal string of the length N can be found in certain evolution processes during N generations under the condition that the total number of fitness function calculations is of the order of N^2. Since the evolution method of search is simple and universal, it can be qualified as a good heuristic optimization method for a large class of problems. In particular, this method can be used to run on-board and on-line for optimizing genetic cloud from swam mode into organism mode, in other word, to obtain coherent collective properties from different genomes. In the next section, the virus-like model will be extended further to demonstrate self-learning and self-adapting features.

5.4.2 Interaction between Evolution and Learning in an Evolutionary Process

The ideas of simplistic evolutionary approach, which improves collective properties of a genetic cloud, can be extended further when to embody informational string into an agent. This approach allows studying not only a collective behavior, but also an interaction between learning and evolution for the collective behavior (Red'ko *et al.*, 2005a), (Red'ko *et al.*, 2005b). This interaction can be observed by using the ACD ANN, described in the introduction to this section. The model and critic ANN can be optimized by both learning and evolution: learning via the reinforcement learning approach (Sutton & Barto, 1998) and evolution via the Darwinian type evolution that includes mutation of ANN synaptic weights and selection of agents with the best control systems.

Description of Agents

Agents are assumed to work in an organism without a common genome, i.e all agents behave independently from each other. Such a behavior can be useful in homeostatic processes, described e.g. in Sect. 2.3.1.1 or in Sect. 4.4. The agent tries to predicts future changes of the homeostatic energetic value, see Sect. 2.3 (or a hormone as suggested in Sect. 4.2) and tries to increase its energetic potential by taking and giving energy from a common bus. Since each agent can exchange and store energy, it is useful to consider "investment" of resources, i.e. when an agent cannot store resources locally, it can invest them into other agents. In this way, each agent has its resource distributed into locally stored ones (so called "cash resources") and invested ones ("stock resources"). The sum of these is the net potential $C(t)$. The agent decision is to modify the variable $u(t)$, which is the fraction of the agent's potential that is currently invested in other agents. The homeostatic behavior of an organism is determined by the time series $X(t), t = 1, 2, \ldots$, where $X(t)$ is the need of resources at the moment t. The higher is the need, the expensive they are for the systems, i.e. the value $X(t)$ is a global cost factor. The goal of the agent is to increase its potential $C(t)$ by changing the value $u(t)$. The potential dynamics is described by

$$C(t+1) = C(t)\{1 + u(t+1)\Delta X(t+1)/X(t)\} \times [1 - J|u(t+1) - u(t)|], \quad (5.7)$$

where $\Delta X(t+1) = X(t+1) - X(t)$ is the current change of the need of resources, and J is a parameter that takes into account losses of energy by transfer through a bus. The factor in the braces corresponds to the change of the potential as the result of a global changes on energy demand. The factor in the square brackets reflects the losses by the energy transfer. For convenience, the logarithmic scale for the agent resource is used, i.e., $R(t) = \log C(t)$. The current agent reward $r(t) = R(t+1) - R(t)$ is defined by the expression:

$$r(t) = \log\{1 + u(t+1)\Delta X(t+1)/X(t)\} + \log[1 - J|u(t+1) - u(t)|]. \quad (5.8)$$

For technological reasons and for simplicity it is assumed that the variable $u(t)$ takes only two values, $u(t) = 0$ (agent switches on a local recharging and taking energy from a bus) or $u(t) = 1$ (local recharging is switched off and giving energy to a bus).

Such a stock-like representation is useful in a genetic cloud, because it allows finding steady stationary states (i.e. homeostasis) without common regulatory elements. Applied to energetic homeostasis, it allows to optimize the global energy distribution and the energy consumption in the organisms by egoistic behavior of virus-like agents.

Agent Learning

As stated previously, the agent control system is a simplified ACD that consists of two ANNs: a model and a critic (see Fig. 5.39). The goal of the adaptive critic is to stochastically maximize the utility function $U(t)$ (Sutton & Barto, 1998):

$$U(t) = \sum_{j=0}^{\infty} \gamma^j r(t+j), \qquad t = 1, 2, \ldots, \quad (5.9)$$

where $r(t)$ is an instantaneous reward obtained by the agent and γ is the discount factor $(0 < \gamma < 1)$. Under the realistic assumption $|\Delta X(t+1)| \ll X(t)$, it is supposed that the ACD state $\mathbf{S}(t)$ at moment t depends on two values, $\Delta X(t)$ and $u(t)$: $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$.

The role of the model ANN is to predict changes of the homeostatic time series. The model output $\Delta X^{pr}(t+1)$ is based on m previous values of ΔX: $\Delta X(t-m+1), \ldots, \Delta X(t)$, which are used as the model inputs. The model ANN is implemented as a multilayer perceptron (MLP) with one hidden layer of tanh nodes and linear output. The operation of this neural network is described by the following expressions:

$$\begin{aligned} \mathbf{x}^{\mathbf{M}} &= \{\Delta X(t-m+1), \ldots, \Delta X(t)\}, \\ y^M{}_j &= \tanh\left(\textstyle\sum_i w^M{}_{ij} x^M{}_i\right), \\ \Delta X^{pr}(t+1) &= \textstyle\sum_j v^M{}_j y^M{}_j, \end{aligned} \quad (5.10)$$

where $\mathbf{x^M}$ is the neural network input vector, $\mathbf{y^M}$ is the vector of outputs of hidden layer neurons, ${w^M}_{ij}$ and ${v^M}_j$ are neuron synaptic weights.

The critic ANN is intended to estimate the state value function $V(\mathbf{S})$ (estimate of U in (5.9)) for the current state $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$, the next state $\mathbf{S}(t+1) = \{\Delta X(t+1), u(t+1)\}$, and its predictions $\mathbf{S^{pr}}_u(t+1) = \{\Delta X^{pr}(t+1), u(t+1)\}$ for two possible actions, $u(t+1) = 0$ or $u(t+1) = 1$. The critic ANN is also a MLP of the same structure as the model. The operation of the critic neural network is described by the following expressions:

$$\begin{aligned} \mathbf{x^C} &= \mathbf{S}(t) = \{\Delta X(t), u(t)\}, \\ {y^C}_j &= \tanh(\textstyle\sum_i {w^C}_{ij} {x^C}_i), \\ V(t) &= V(\mathbf{S}(t)) = \textstyle\sum_j {v^C}_j {y^C}_j, \end{aligned} \tag{5.11}$$

where $\mathbf{x^C}$ is the neural network input vector, $\mathbf{y^C}$is the vector of outputs of hidden layer neurons, ${w^C}_{ij}$ and ${v^C}_j$ are neuron synaptic weights.

At any moment t, the following operations are performed:

1. The model ANN predicts the next change of the time series $\Delta X^{pr}(t+1)$.
2. The critic ANN estimates the state value function for the current state $V(t) = V(\mathbf{S}(t))$ and the predicted states for both possible actions ${V^{pr}}_u(t+1) = V({\mathbf{S}^{pr}}_u(t+1))$, where ${\mathbf{S}^{pr}}_u(t+1) = \{\Delta X^{pr}(t+1), u(t+1)\}$, and $u(t+1) = 0$ or $u(t+1) = 1$.
3. The ε-greedy rule (Sutton & Barto, 1998) is applied: the action corresponding to the maximum value ${V^{pr}}_u(t+1)$ is selected with probability $1-\varepsilon$, and an alternative action is selected with probability $\varepsilon (0 < \varepsilon \ll 1)$. (For example, if

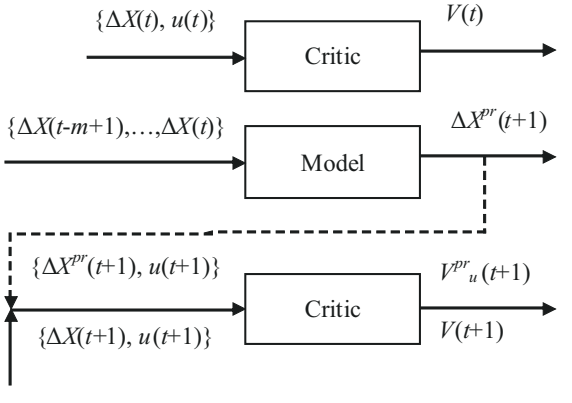


Fig. 5.39 The scheme of the considered adaptive critic design (ACD). The ACD consists of two neural networks: a model and a critic. The model ANN predicts changes of the time series. The critic ANN (the same ANN is shown in two consecutive moments) forms the state value function for the current state $\mathbf{S}(t) = \{\Delta X(t), u(t)\}$, the next state $\mathbf{S}(t+1) = \{\Delta X(t+1), u(t+1)\}$, and its predictions $\mathbf{S^{pr}}_u(t+1) = \{\Delta X^{pr}(t+1), u(t+1)\}$ for two possible actions, $u(t+1) = 0$ or $u(t+1) = 1$.

the action $u(t+1)=0$ corresponds to the maximum of ${V^{pr}}_u(t+1)$, then it is selected with probability $1-\varepsilon$; alternatively, the action $u(t+1)=1$ is selected with probability ε.).

4. The selected action is carried out. The transition to the next time moment $t+1$ occurs. The current reward $r(t)$ is calculated in accordance with (5.8) and received by ACD. The value $\Delta X(t+1)$ is observed and compared with its prediction $\Delta X^{pr}(t+1)$). The weights of the model ANN are adjusted to minimize the prediction error by means of the usual method of error backpropagation (Rumelhart *et al.*, 1986) and the gradient descent with $\alpha_M > 0$ as the model learning rate.
5. The critic computes $V(t+1)$. The temporal-difference error (Sutton & Barto, 1998) is calculated:

$$\delta(t) = r(t) + \gamma V(t+1) - V(t). \tag{5.12}$$

6. The weights of the critic ANN are adjusted to minimize the temporal-difference error (5.12) using its back propagation and the gradient descent with $\alpha_C > 0$ as the critic learning rate.

Evolution of Agents

The Darwinian evolution of agent populations is considered. An evolving population consists of n agents. Each agent has a resource $R(t)$ that changes in accordance with values of agent rewards: $R(t+1) = R(t) + r(t)$, where $r(t)$ is calculated in (5.8). Evolution passes through a number of generations, $n_g = 1,2,\ldots$.The duration of each generation n_g is T time steps (agent lifetime). At the beginning of any generation, initial resource of each agent is zero, i.e., $R(T(n_g-1)+1) = 0$.

The initial synaptic weights of both ANNs (model and critic) form the agent genome $\mathbf{G} = \{\mathbf{W}_{M0}, \mathbf{W}_{C0}\}$. The genome $\mathbf{G}$ does not change during the agent life and it is fixed when the agent is born. However, synaptic weights of the ANNs $\mathbf{W}_M$ and $\mathbf{W}_C$ are changed during agent life via learning described above.

At the end of each generation, the agent having the maximum resource $R_{max}(n_g)$ is determined (the best agent of the generation n_g). This best agent gives birth to n children that constitute a new $(n_g+1)-$th generation. The children genomes $\mathbf{G}$ differ from their parent genome by small mutations taken from a normal distribution.

At the beginning of every new $(n_g+1)-$th generation, we set for each agent $G_i(n_g+1) = G_{best,i}(n_g) + rand_i$, $\mathbf{W}_0(n_g+1) = \mathbf{G}(n_g+1)$, where $\mathbf{G_{best}}(n_g)$ is selected from the best agent of the previous n_g-th generation, and $rand_i$ is $N(0, {P^2}_{mut})$, i.e., a normally distributed random number with zero mean and standard deviation P_{mut} (mutation intensity), which is added to each synaptic weight.

Thus, the genome $\mathbf{G}$ changes only via evolution, whereas the synaptic weights $\mathbf{W}$ are adjusted only via learning.

General Features of Agent Learning and Evolution

Two examples of the time series $X(t)$ are considered: sinusoidal (with the period of 20 time moments) and smoothed stochastic time series. The simulation parameters are as follows: discount factor $\gamma = 0.9$, number of inputs of the model neural network $m = 10$, number of hidden neurons of the model and critic $N_{hM} = N_{hC} = 10$, learning rate of the model and critic $\alpha_M = \alpha_C = 0.01$, parameter of the ε-greedy rule $\varepsilon = 0.05$, mutation intensity $P_{mut} = 0.1$, population size $n = 10$. Generation duration T is specified below. Agent expenses are neglected ($J = 0$) in the main part of simulations.

The following cases are analyzed:

- **Case L** (pure learning); i.e., a single agent that learns by means of temporal difference method, see Eqs. (5.8), (5.9), (5.12);
- **Case E** (pure evolution), i.e., evolving population without learning;
- **Case LE**, i.e., learning combined with evolution, as described above.

The results are illustrated by Fig. 5.40, where maximal resource values attained by agents during 200 time steps for these three cases of adaptation are shown. For the cases E and LE, the maximal value of agent resource in a population $R_{max}(n_g)$ at the end of each generation n_g is recorded; generation duration T is equal to 200. For the case L, just one agent whose resource is reset $R(T(n_g - 1) + 1) = 0$ after the passing of every $T = 200$ time steps is considered; the index n_g is also incremented by one after every T time steps. The plots R_{max} vs. n_g for the sinusoidal time series are shown in Fig. 5.40. In order to exclude the decrease of the value $R_{max}(n_g)$ that is due to the random choice of actions when applying the ε-greedy rule for the cases LE and L, it is set $\varepsilon = 0$ after $n_g = 100$ for the case LE and after $n_g = 2000$ for the case L. The results are averaged over 1000 simulations.

Analysis of agent actions demonstrates that both cases E and LE ensure the finding of optimal policy, i.e. the agent performs the action $u(t+1) = 1$ or $u(t+1) = 0$ at prediction of rises or falls of the time series $X(t)$ (see the curves E and LE in Fig. 5.40). With this policy, the agent attains asymptotic value $R_{max} = 6.5$. The pure learning is able to find only the satisfactory policy (see the curve L), namely, the agent performs the action $u(t+1) = 1$ when the time series $X(t)$ rises (or falls by a small amount) and performs the action $u(t+1) = 0$ when the time series $X(t)$ falls significantly. The asymptotic value of R_{max} for the case L is only 5.4.

The described results outline main features of modeled processes for the sinusoidal time series. Similar learning and evolutionary processes are observed for smooth stochastic time series (Red'ko *et al.*, 2005b).

Interaction between Learning and Evolution: Baldwin Effect

So, pure learning is imperfect; nevertheless, learning helps evolution to attain larger values of R_{max} faster (see the curves E and LE in Fig. 5.40). Some other simulations (at rather large generation durations) also confirm that learning helps evolution to

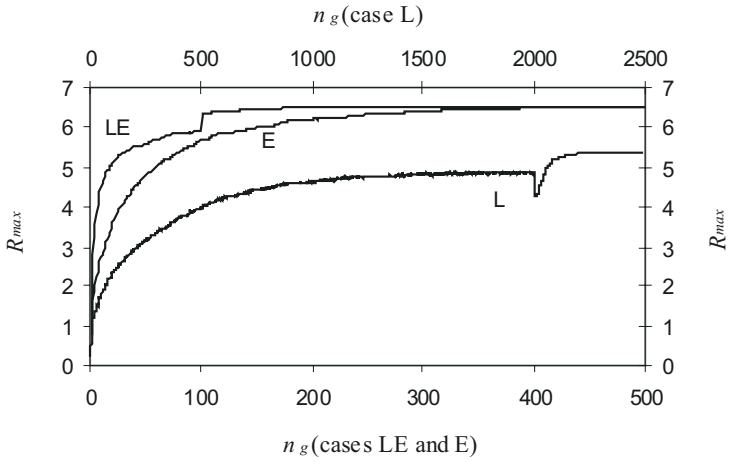


Fig. 5.40 The plots of $R_{max}(n_g)$ for the sinusoidal time series. The curves LE, E and L correspond to the cases of learning combined with evolution, pure evolution and pure learning, respectively. $T = 200$. Results are averaged over 1000 simulations.

find a good policy. For example, Fig. 5.41 demonstrates that during the early generations ($n_g = 1 - 2$) a satisfactory policy is found via learning (only after 200-300 time steps of the agent life).

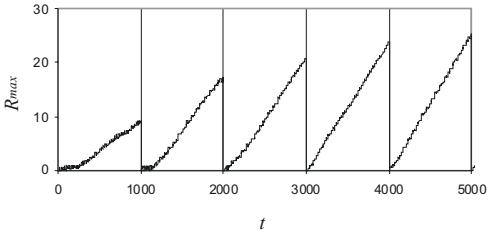


Fig. 5.41 The plots of the resource $R_{max}(t)$ of the best agent in the population for the first five generations, see description in text.

This is the case of learning combined with evolution on the sinusoid time series; generation duration $T = 1000$. The ends of generations are shown by vertical lines. During early generations (generations 1 and 2), there is an obvious delay in the increase of the agent resource. An advantageous policy is found only after some learning period during first 200 to 300 time steps. By the fifth generation, the rapid increase of the resource begins at the start of the generation, demonstrating that the advantageous policy has become inherited. In this generation, the agents exhibit a satisfactory policy from the beginning of the generation, and the learning does not improve the policy significantly. Thus, we can see that the initially learned policy becomes inherited; this corresponds to the Baldwin effect (Baldwin, 1896), (Turney *et al.*, 1996).

Thus, the model demonstrates that the evolutionary optimization can be more effective, than training with the reinforcement learning. The behavior of simulated

agents is interesting; in particular, nontrivial genetic assimilation of acquired features in Darwinian evolution takes place.

5.4.3 Evolutionary Emergence of a Cooperation between Agents

The previous section was devoted to a homeostatic behavior of purely software virus-like agents in a genetic cloud. In this section we consider an aggregation behavior of the same agents, which however are embodied into a physical robot (Kernbach *et al.*, 2009a) and demonstrate evolutionary emergence of cooperation (Burtsev & Turchin, 2006). Each physical robot can have several such agents, running as separate processes on MPU and being in charge of different activities. In this context agents are "virtual" in contract to real robots. A robot can be "dead"; this basically means an "empty" physical body, which can be occupied by different virtual agents. An agent can reproduce itself in the same physical robot or it can be transferred to such "empty" robots. We can imagine that initially there are more "empty" robots, than virtual agents and during evolution, population of virtual agents occupies more and more physical bodies, see also Sect. 5.3 about artificial sexuality. In the following we assume that only one virtual agent occupies one physical robot.

A population of autonomous agents evolves in two-dimensional plane, see Fig. 5.42. Each robot normally occupies some area, which can be considered as a "cell". Each such "cell" contains a robot or is empty. A robot can occupy an empty area with a certain probability per time step. An area becomes empty when two robots are aggregated (any "cell" can contain several agents) or a robot moves away. Agents interact each with others, in particular, any agent can compete with other agents for resources. As in the previous section, the control system of agents is an ANN, which connects sensors to motors. The operation of the neural network is described by the expression:

$$O_j(t) = \sum_i w_{ij} I_i(t),\ t = 1, 2, \ldots, \tag{5.13}$$

where $\mathbf{I}(t)$ and $\mathbf{O}(t)$ are input and output vectors of the neural network, w_{ij} are neuron synaptic weights. The input vector $\mathbf{I}(t)$ provides information about the presence of resource in the field of vision, the level of internal resource and the Euclidean distance between marker vectors of the agent and its partner for potential interaction. At each time step t, the agent performs the action associated with the maximum output value $O_j(t)$.

Agents can do nothing (rest), finding energy source (eating), aggregate, produce a virtual offspring (divide), go forwards (move), make a turn to the left or right (turn), and compete with another agent (fighting). The agent spends its energy resource at any action; the resource is increased when aggregating (by taking energy from a common bus) or when finding energy source for individual recharging. If agent resource is completely depleted, the agent virtually dies. The least energetically demanding action is rest, the most demanding is fighting. When an agent divides, one virtual offspring is created and placed in the same cell as the parent; the parent

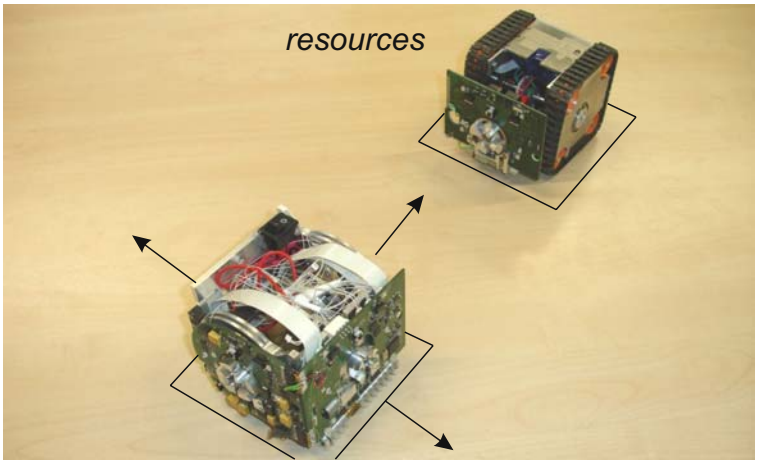


Fig. 5.42 The agent in two-dimensional world. Agents can rest, looking for resources (e.g. other robots for aggregate), divide, move forward, turn to the left or right, and compete with other agents.

then transfers half of its energy to the offspring. When one agent hits another, the victim loses an amount of energy resource, which is transferred to the attacker.

Sensory inputs of agents include its energy resource and how many other agents are in the field of vision, see Fig. 5.42. Each agent has external phenotype that is coded by a vector of integer values (marker). The markers do not influence agent behavior but function only as indicators of similarity. The Euclidian distance between an agent's marker and the marker of another agent is also a sensory input of the agent neural network. Both weights of the neural network and the marker are inherited by the offspring when an agent divides.

The genome of the agent codes its neural network (the neural network structure and synaptic weights of neurons). Additionally, the agent genome includes the marker that is the gene of similarity. This marker is obtained by the agent from its parent; so the marker difference for the offspring and the parent is small, because this difference is created only by small mutations. Agents are able to perceive markers of neighboring agents. An evolution of an agent population is considered; neural network structures and neuron synaptic weights are modified during genome mutations. Behavioral strategies of agents are not predetermined; instead, the process of evolution constructs and reconstructs them from elementary actions. Agents of an initial population are unaware of markers. Thus, the use of markers in a population has to emerge during the evolution.

Analysis of the model without markers (Burtsev & Turchin, 2006) shows that the strategies evolving in the simulation correspond to those in the well-known game of dove-hawk-bourgeois (Smith, 1974). Doves never attack other agents and attempt to escape when attacked, whereas hawks make a living by predation on other agents. The bourgeois strategy in the considered model is to stay in the same cell and immediately attack any invader, while ignoring agents in neighboring cells (unlike the

hawks). In the model without markers the dominant strategy is bourgeois, provided that the energy resource is sufficiently large.

In the full model, in which agents can evolve the ability to detect phenotypic similarity (distance between the markers), three kinds of cooperative strategies emerge. The first one is simply the cooperative version of the dove. Cooperative doves ignore out-group (phenotypic distance is large) members, but leave cells containing within-group (phenotypic distance is small) members to avoid competing with them. In the second strategy, agents also leave cells containing within-group members, but when they detected out-group members they attacked them. This strategy can be called "raven", corresponding to the Russian proverb, "a raven will not peck out the eye of another raven". The third cooperative strategy is to stay in the same cell containing within-group members and collectively fight with any out-group invader. Having to share limited energy of the cell means that agents using this cooperative defense strategy have small internal energy resource, but they still have a good chance of defeating a large invader because of their advantage in numbers. This strategy resembles the "mobbing" behavior that many species of small birds, such as starlings, use to drive away large predators. For this reason, this strategy is called the "starling" strategy.

Thus, in the absence of phenotypic markers, three distinct strategies emerge. These strategies correspond to the dove, the hawk, and the bourgeois. In the presence of markers, the evolution results in some predictable modifications of these basic strategies, but also in the emergence of a new one. Cooperative doves avoid competition with in-group members, whereas cooperative hawks - "ravens" - avoid attack on phenotypically similar agents. The new strategy is the starlings, which live in groups and defend living territory cooperatively against predation. These results indicate that cooperative strategies can evolve even under certain minimalist assumptions, provided that agents are capable of perceiving heritable external markers of other agents. In should be noted that the cooperation emerges internally in evolving population, during an evolutionary self-organization.

Evolutionary Disappearance of Fighting Genes

The early version of this model does not consider similarity genes (Burtsev, 2002). Nevertheless, disappearance of fight effectors from agent control systems during evolutionary processes is observed. Similar to the model described above, an evolving population of embodied autonomous agents is considered. An agent can rest, eat, divide, move forward, turn to the left or right, and fight with another agent. Sensors and effectors can be removed and restored via mutations of agent neural networks. The main difference with the model described above is absence of agent markers. The interesting phenomenon in this model is observed, namely, surprising peaks in population size N, see Fig. 5.43, the curve below.

In order to analyze this phenomenon, fight effectors are externally excluded from the agent control systems. In this case population size $N(t)$ is significantly increased, see Fig. 5.43, the curve above. Thus, the observed peaks in the full model are due to disappearance of fight effectors from agent control systems via mutations during

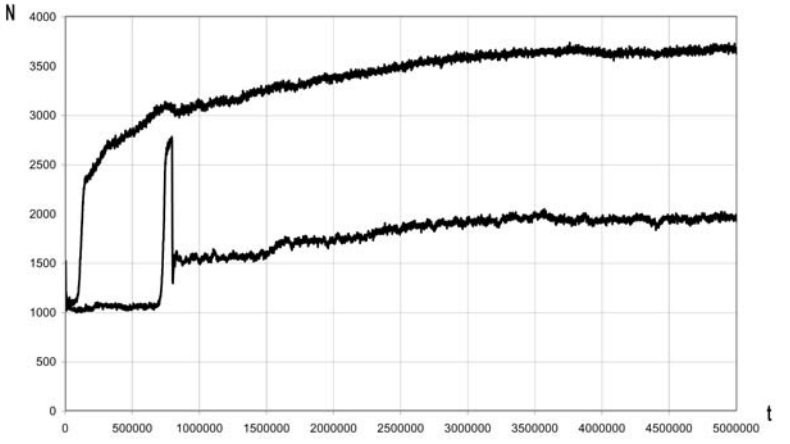


Fig. 5.43 Time dependence of population size $N(t)$ in the full model (curve below) and after externally removing of fight effectors from agent control systems (curve above).

evolutionary processes. These fight effectors are removed from control systems of all agents of the population. So, agents do not fight each with others, instead they use their energy resource for other useful actions. It should be noted that duration of peaks of population size is small: fighting effectors are restored via mutations and the population size is decreased.

5.4.4 Discovering of Chains of Actions by Self-learning Agents

In this section the described above virus-like agent in a genetic cloud is slightly modified: agent does not have markers, any "cell" can contain only one agent, the agent control system is a set of logical rules instead of a neural network. The logical rules of an agent have the form "if situation $\mathbf{S}(t)$ is present, then the action $A(t)$ should be executed", where t is time moment. All actions and other behaviors are equal to the described in the previous section. The evolving population consists of n agents. Any agent has an energy potential $R(t)$ that is increased by recharging and decreased at executing of actions by the agent, for $R(t) = 0$ an agent is dead. The aim of this models is to demonstrate that agents are able to discover chains of actions leading to replenishment of $R(t)$. In other words, virus-like agents can demonstrate a self-adapting behavior, which increase their surviving chances.

The agent control system is a set of logical rules:

$$\mathbf{S}_k(t) \rightarrow A_k(t), \tag{5.14}$$

where $\mathbf{S}_k(t)$ is the current situation, $A_k(t)$ is the action corresponding to this rule. Each rule has the weight W_k that is modified at agent learning. Components of the vector $\mathbf{S}_k(t)$ are equal to 0 or 1. Values 0 or 1 correspond to absence or presence of energy or another agent in "the field of vision" of the agent. The set of logical rules $\{\mathbf{S}_k(t), A_k(t), W_k\}$ constitutes the genome of the agent.

Each time step t any agent selects an action and is learned. Additionally, analog of annealing method (Kirkpatrick *et al.*, 1983) is used: at initial time moments, when set of logic rules of agents is generated, actions are selected randomly with probability $\varepsilon \sim 1$, then the value ε is gradually decreased to zero; at further time moments, the agent uses the rule that has maximal weight W_k among rules corresponding to the current situation $\mathbf{S}_k(t)$.

At learning weights of rules W are adjusted by reinforcement learning (Sutton & Barto, 1998). The change of the weight of the rule used at time moment t-1 is determined by the following expression:

$$\Delta W(t-1) = \alpha[R(t) - R(t-1) + \gamma W(t) - W(t-1)] \,, \tag{5.15}$$

where $W(t-1)$ and $W(t)$ are weights of rules applied at time moments $t-1$ and t, $R(t) - R(t-1)$ is the change of agent resource at time transition from $t-1$ to t, α is the parameter of the learning rate, γ is the discount factor.

Several Results

Simulation is performed for the full described model and for the simplified version of the model. In the simplified version only one agent is studied. In case of the full version of model, the population consists from 50 agents, the two-dimensional world contains 100 cells, energy sources are randomly distributed in 50 cells. The parameters of simulation are as follows. The decrease of agent energy resource at all actions (except fighting) is 0.01. At fighting the decrease of agent energy resource is 0.02; the amount of energy resource that is transferred to the attacker from the victim is 0.05. The increase of agent energy by recharging is 1.0. Parameters of reinforcement learning are $\alpha = 0.1, \gamma = 0.9$. The annealing method is used as follows. At $t = 0$ it is set $\varepsilon = 1.0$, then the probability of random choice of action ε is exponentially reduced to zero, characteristic time of reduction ε is equal to 1000 time steps. The initial energy resource of the agent (at $t = 0$) is $R(0) = 1.0$.

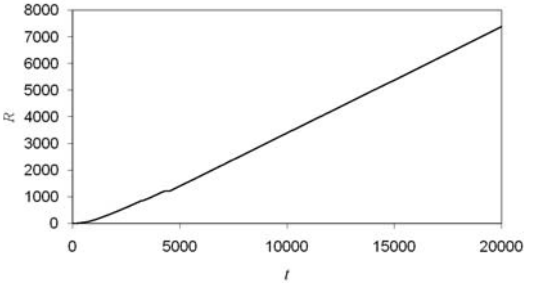


Fig. 5.44 Time dependence of the agent energy resource $R(t)$.

Simulations for the case of the full version of model demonstrate that rather natural behavior of autonomous agents is formed via learning and evolution. The main actions of agents are "to eat" and "to fight"; these actions lead to increase of energy resource of the agent. Agents avoid the action "to divide" which leads to resource reduction. Each of other actions is executed with small frequency.

In the simplified version of model a single autonomous agent is analyzed. The set of agent rules is formed by means of reinforcement learning. The agent can execute 5 actions: to eat, rest, move forward and turn to the right or left. The main parameters of simulations are equal to parameters of the full model. Simulations demonstrate that in this case chains of actions (unknown to the agent in advance), leading to recharging by the agent are formed.

The example of time dependence of the agent resource $R(t)$ is shown in Fig. 5.44. Each situation **S** is characterized by presence/absence of energy in 4 "cells" (4 bodylength) of a field of vision. So, there are 16 possible situations **S**. Taking into account that there are 5 possible actions, we have that there are 80 possible rules. Note that in any simulation at initial time steps during random search all these possible rules are formed. However, weights of these rules modified in the course of reinforcement learning are different.

The essential logic rules that have large weights (greater than 1.0) at the end of simulation are selected. The number of these selected rules for typical simulation is equal to 16; each rule corresponds to the certain situation **S** and the action executed in this situation. Just these rules are applied by the agent. This set of rules can be considered as the heuristics, formed by the agent in the course of self-learning. These of heuristics are as follows. 1) If energy is located in the same cell, where the agent is present, then the agent executes the action "eat"; this corresponds to the simple one-link chain of actions. 2) If energy is not present in the agent cell and there is the energy in the cell forward to the agent or in the right/left cell, then the agent executes the action "to move forward" or "turn to the right/left"; this corresponds to the two-link or three-link chain of actions, respectively. These chains consisting from one/two/three actions result in energy consuming at the end of a chain. Additionally, if there is no energy in the agent field of vision, the agent prefers the action "to move forward". This can be explained to the fact that moving forward corresponds to the two-link chain, whereas turning corresponds to the three-link chains. Note that the action "to rest" is ignored in all simulations. In some simulations there was a small number other rules having large weights; nevertheless, properties of rules applied by an agent can be only slightly differed from described above. The dependence $R(t)$ presented in Fig. 5.44 corresponds to 16 selected rules described above. So, in the course of self-learning the agent forms quite natural rules defining reasonable strategy of behavior.

Thus, the chains of actions are discovered by self-learning autonomous agents. The result of these chains is energy recharging and increasing of energetic potential of the agent. These chains are unknown to the agent in advance; they are formed by means of reinforcement learning in accordance with (5.15). It should be noted that the current model can be developed further in two interesting directions. The first important direction of the future researches is more detailed analysis of behavior of agents that have several needs, for example, needs of energy, reproduction, and safety. It is natural to suppose that there is certain competition between needs. Any need can be accompanied with the corresponding motivation $M(t)$. Dynamics of a motivation can be modeled similar to the work (Nepomnyashchikh *et al.*, 2008) that takes into account the inertia of motivation change, random variations and the

directed change of the motivation $M(t)$. Competitions between motivations and needs could be also analyzed. The second direction of further research is inclusion of predictions of next situations $\mathbf{S}(t+1)$ into the model. It is possible to estimate the sum of future energy resource increase for situations (states) $\mathbf{S}(t)$ and execute that actions which results in situations $\mathbf{S}(t+1)$ corresponding to maximal estimated resource increase. It is natural to use agent resource change $R(t) - R(t-1)$ as a component of situation vector $\mathbf{S}(t)$. Such version of the model can be similar to the model of autonomous agents described in Sect. 5.4.2.

Additionally, we can note that forming chains of actions corresponds actually to generalization of processes of agent behavior. Namely, only 5 heuristics are used, whereas the whole agent control systems have 80 logical rules. Generation of these heuristics can be considered as origin of some form of semantics of agent behavior. Moreover, using such predictions and heuristics, an agent can produce "inferences" and some "logic" of adaptive behavior. So, basing on this approach, we can study evolutionary steps of creating adaptive behavior and appearance of certain "logical conclusions". This process is supposed to be similar not only to simple virus-like organisms, but also to more higher animals with cognitive capabilities (Red'ko, 2008).

5.4.5 Virus-Like Organisms: New Adaptive Paradigm ?

This chapter[16] introduced a new paradigm towards self-adapting systems with cloud genotype and structurally flexible phenotype. This concept may be especially useful for artificial organism during the swarm mode, initial aggregation and several self-regulation phases, see Table 1.4. Since in the literature such features are usually related to viruses, we proposed virus-like organization of artificial organisms for these activities. It is demonstrated that evolutionary process in cloud genotype (i.e. quasispecies model) provides a good heuristic optimization properties, which can be used for on-line and on-board evolving of controllers.

Several such controllers, denoted as agents, are considered. These agents are supposed to run in organism's computational system, as described in Sect. 5.4.2 and perform homeostatic activities, or be physically embodied, as indicated in Sect. 5.4.3 and execute behavioral tasks. These agents are based on ANN and logical rules, where the behavior is optimized by means of both reinforcement learning and evolutionary optimization. It is shown that evolutionary optimization can be more effective as compared with reinforcement learning that includes random search. The evolutionary emergence of cooperation between agents is described. It is demonstrated that in addition to well-known strategies of dove, hawk, and bourgeois, several forms of colony-like cooperative strategies evolutionary emerge, e.g. the "starling" strategy, when a family of virus-like agents collectively fights with any foreign invader. Moreover, such agents can demonstrate self-learning properties, leading

[16] This work is partially supported by the Russian Foundation for Basic Research, Grant No 07-01-00180 and the Program of the Presidium of the Russian Academy of Science "Intelligent informational technologies, mathematical modeling, system analysis and automatics", Project No 2.15.

to adaptive behavior in unknown situations. The chains of actions can be thought as of a generalization of input information by the agent and formation of certain forms of semantics. This model can be further developed in order to analyze how a genetic cloud can use certain "logical collective conclusions" for achieving useful results.

Based on this work, we assume that genetic cloud can provide certain collective properties related to optimization, self-learning and adaptive collective behavior. It is worth to look more carefully for adaptive properties of real viruses and virus colonies to utilize their high adaptivity for artificial systems.

5.5 Towards the Emergence of Artificial Culture in Collective Robotic Systems

Alan Winfield, Frances Griffiths

This chapter explores the possibility that we might be able to engineer a robot collective ("society" of robots) in which we observe the emergence of "cultural" traditions or traits: an artificial, robot culture. However, this is not merely a *gedanken* experiment: a current project is presented here with the title "the emergence of artificial culture in robot societies", which aims to demonstrate — or at least take the first steps toward demonstrating — behaviours that can be argued as evidence for emerging cultural traits in a society of real robots. We first outline the aims and inspiration for the project, then describe the experimental infrastructure (artificial culture lab). Then follows a discussion of the significant challenges the project faces in observation and interpretation of experiments within the artificial culture lab, followed by a consideration of robot memes and how meme evolution might be tracked.

5.5.1 Project Aims

The Artificial Culture project aims to address and illuminate the question "how can culture emerge and evolve as a novel property in groups of social animals?" by building an artificial society of embodied intelligent agents (real robots), creating an environment (artificial ecosystem) and appropriate primitive behaviours for those robots, then free running the artificial society. The aims of the project lie primarily in modelling the processes and mechanisms by which a group of embodied agents (robots) might make the transition from social to cultural. Even with small populations (a few tens) of relatively simple robots we see, in a short time, a very large number of interactions between robots. Compared with computer simulated robots the inherent heterogeneities of real robots, and the noise and uncertainty of the real world, vastly increase the space of possibilities and the scope for unexpected emergence in the interactions between robots.

In the project we are attempting to create the conditions and primitives in which proto-culture can emerge in a robot society. Robots will, for example, be able to copy each other's behaviours and select which behaviours to copy. Dawkins coined

the term "meme" to describe a unit of cultural transmission (Dawkins, 1976), and we use this terminology here. Imitated behaviours (memes) will mutate because of the noise and uncertainty in the real robots' sensors and actuators, and successful memes will undergo multiple cycles of copying (heredity), selection and variation (mutation). Furthermore we will explore a bi-phased approach in which we alternate between real-time (with real physical robots) in which the emergence, selection and refinement of these discrete behavioural artefacts (memes) takes place; with evolutionary time, in which we run a genetic algorithm (GA) process to grow and evolve the robots' controllers so that the behaviours and premiums associated with the emerging memes become hard-wired into the robots' controllers.

The project is inspired by the *Copybots* suggested by (Blackmore, 1999), pages 106-107, and by Dautenhahn's 1995 paper "Getting to Know Each Other - Artificial Social Intelligence for Autonomous Robots" (Dautenhahn, 1995). From a technical perspective the project draws upon a multi-disciplinary body of literature in imitation (Nehaniv & Dautenhahn, 2007); for instance the work of (Alissandrakis *et al.*, 2007) describing imitation leading to "cultural transmission of behaviours and emergence of proto-culture" between two simulated 2D two-jointed robotic arms. However, we argue that a multi-robot "society" is a necessary substrate for this work and bring key concepts from the field of swarm robotics (Beni, 2005). Furthermore, our robots need to combine social learning with evolution, and we thus aim to integrate techniques from evolutionary robotics (Nolfi & Floreano, 2000a; Trianni, 2008). We believe this project to be the first to attempt meme-gene co-evolution with multiple real robots.

5.5.2 The Artificial Culture Laboratory

Core to the project is the creation of an artificial environment: the artificial culture lab. The artificial culture lab comprises a physical space ("arena") designed for and populated by miniature wheeled mobile robots. The arena is closed in the sense that its physical boundaries define the edges of the robots' world, out of which they cannot physically stray. The arena is not hermetically sealed, thus robots (since they have both light and sound sensors) are affected by ambient lighting or noise levels. Providing that these external environmental influences do not overwhelm (blind) the robots' sensors, they are not a problem. Indeed, a certain level of background noise in the environment is considered essential as it will contribute to imperfect robot-robot imitation or communication, and changing light levels (day and night) may be useful in providing the robots with a circadian rhythm.

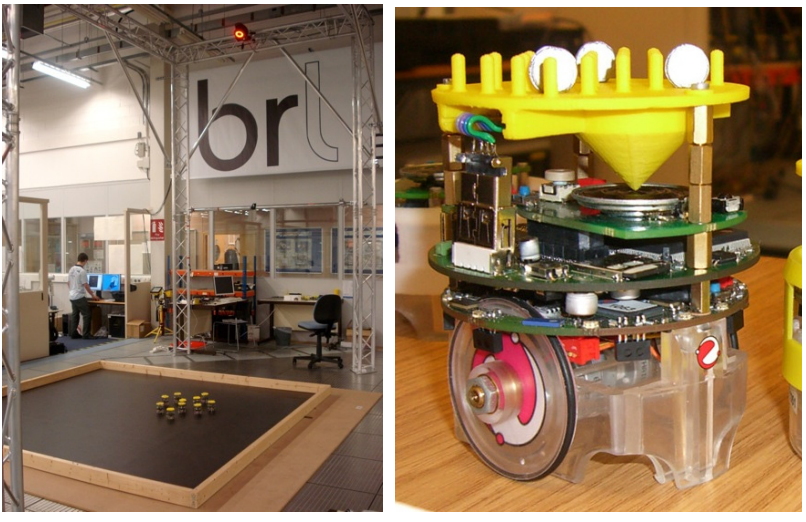
We will divide the population of robots into physically separate, but not isolated, sub-groups or "villages". Each village consisting of say 5-10 robots, with perhaps four or five villages in the arena. This is to allow for the possibility of the parallel emergence and evolution of different memes or meme complexes in the (relatively) separated villages, and the potential for emergent meme trading between villages. We also believe that the "village" approach will encourage diversity in both gene- and meme-pools across the whole population. We will encourage this diversity by

providing slightly different artificial ecosystems within each village. The ecosystems will be necessarily simple (because the robots are simple) but various possibilities present themselves: for instance resources (energy) will be represented by active artefacts (small beacons with sound or light emitters). Another possibility is that of other robots programmed to behave as prey. Environmental pressures could then be introduced: famine for instance.

The robots, called e-pucks, are wheeled, differential-drive, robots capable of moving forwards or in reverse, or turning (including turning on the spot) (Mondada *et al.*, 2009). They are equipped with a range of sensors, including short-range infra-red and/or ultra-sound proximity and ranging sensors that allow the robots to sense the presence, direction and range of obstacles and other robots close by. Importantly, robots can sense and track the movements of other robots nearby (albeit imperfectly because of their limited sensors); thus robots have the physical means for imitation. They have multi-coloured programmable lights (LEDs), and simple cameras; microphones and speakers. We have a wide range of options for robot-robot interaction. Robots can signal to each other with movement, light, or sound, one-to-one or one-to-many, and with or without active consent (i.e. one robot can eavesdrop on the communication between two others). The robots are not equipped with manipulators (grippers), thus the only way they can physically act upon the world is with their own bodies (i.e. by pushing light objects, or co-operating with other robots to push heavier objects).

The artificial culture lab is fully instrumented. A tracking system allows the movements of all robots to be captured and recorded for analysis and interpretation. Wireless communication with each robot allows data logging, allowing the emerging memotypes to be captured for analysis. Webcams provide video capture for analysis, and importantly video for project web-pages for open access to support interpretation. Fig. 5.45(a) shows the artificial culture lab in the Bristol Robotics Lab (BRL); Fig. 5.45(b) shows one of the e-puck robots fitted with Linux extension board and tracking "hat".

The use of real physical robots in an artificial ecosystem as described above, rather than computer simulated agents, is central to the methodology proposed for this project. The rationale is that real robots provide vastly more scope for emergence in their interactions than simulated agents. The combination of imperfect sensors; sensing errors that occur because of the distance between robots; multiple robots sharing the same environment (i.e. occlusion of robots by each other) and sharing the same communications modality (i.e. all talking at once); small differences between sensors and actuators (motors) which mean that the robots are not all identical; real-world physics which means that each experimental run (even with the same starting conditions) will quickly diverge into a new space of possibilities; noise in the environment and, unexpected non-fatal faults (i.e. a faulty wheel which gives the robot a "limp"), we argue could not be created in simulation (to do so each of the factors listed would have to be separately modelled, and those models would inevitably lead to simplification thus chronically limiting the space of possibilities). Even in the designed artificial environment we propose here, the use of real physical robots provides vast scope for unexpected emergence. Thus, we argue, behavioural



(a) (b)

Fig. 5.45 (a) Artificial culture lab showing about 10 robots in the arena. (b) An e-puck with Linux board fitted in between the e-puck motherboard (lower) and the e-puck speaker board (upper). Also note the yellow "hat" which here serves three different functions: (1) it provides a matrix of pins for the reflective spheres which allow the tracking system to identify and track each robot; (2) it provides a mounting for the USB WiFi cardwhich slots in horizontally (the wires connecting to the WiFi card are above the USB connector); and (3) it provides an inverted cone to reflect sound from the e-puck's speaker horizontally so that it can be heard by other e-pucks.

artefacts that might be interpretable as artificial memes – elements of an artificial proto-culture – will emerge for no other reason than that they can.

5.5.3 The Challenges and the Case for an Emerging Robot Culture

The project outlined here has clearly made some very significant assumptions about the pre-requisites for the emergence of culture. In essence we have assumed two basic requirements in the embodied agents of the Artificial Culture Lab: social learning and artificial evolution. Furthermore the social learning, through imitation, is designed to undergo a Darwinian process so that behavioural artefacts (memes) evolve within the population. Not least because it lends itself to an empirical approach, we accept – as a working hypothesis – the memetic theory of cultural evolution (Blackmore, 1999), and we consider the initiation of gene-meme co-evolution to be a key underlying mechanism in the transition from no culture to proto-culture.

Assuming that our starting assumptions are not fatally flawed the project still presents significant technical as well as philosophical challenges, as follows.

(a) Following (Richerson & Boyd, 2001) culture as an adaptation requires the correct dynamics between individual and social learning, and genetic evolution: there must be environmental variation — not in a single agent's lifetime but over (say) tens of generations. Given that the emergence of culture is probably highly contingent then, even if we do have the right artificial ecosystem, initial conditions and dynamics, we have no guarantee that robot culture will emerge within the lifetime of the project.
(b) Even if interesting behavioural artefacts do emerge how can we make sense of what we observe, given that it will be robot- rather than human-culture, and what tests must we apply in order to make any robust claims that the behaviours observed really are evidence of an emerging robot proto-culture, rather than merely social behaviours?
(c) If we believe we do see evidence of the emergence of a robot proto-culture then is what we have learned about how that proto-culture has emerged, generalisable from robots to humans (or indeed any social species)? In other words, could there be a universal theory of cultural evolution?

Let us focus on the hermeneutic problem, (b) above. We address this challenge in two ways. Firstly, by proposing a definition of artificial culture as follows:

> sustained and measurable emerging differences in behaviour between two or more groups of robots, where those groups have divided or split off from a common ancestral group, and the behaviours are traceable to common root behaviours in the ancestral group.

This definition has the merit that it is differential and measurable it essentially encodes emerging different traditions between groups. If we accept Whiten's view of cultures as "defined by multiple traditions" (Whiten *et al.*, 2007), then if we observe the emergence of a number of sustained traditions we should be able to label these, with some confidence, as evidence of artificial culture. We expect to recognise these traditions through noticing differences in the behaviour of robots in different groups, when engaged in similar actions. Within the capability of the robots it is possible that robots in one group might, for instance, configure themselves in a particular pattern when co-operating to push an object, whereas robots in another group configure themselves differently when pushing a similar object. Both might be equally effective in moving the object although the efficiency (however evaluated) may differ. Another example may be that one group of robots emits a particular sound within an interaction (perhaps picked up during imitation from an extraneous noise) whereas another group does not (perhaps they were not imitating at the time of the extraneous noise or they were positioned such that their sensors did not pick it up — there could be many reasons for the origin of the difference). An important point here is that our approach does not rely on some subjective judgement of what constitutes "cultural" and hence avoids the trap of anthropomorphism. The necessary corollary is, of course, that the robot traditions — the artificial cultures we measure — will

not make sense from a human cultural perspective. But why should they? This is robot not human culture and essentially alien: an exo-culture.

However, we do not completely reject the subjective, and the second strand of our approach to mitigating the hermeneutic challenge will be to employ an open science approach. Our thinking here is that human beings are very good at identifying the emergence of new patterns, even within complex structures. By placing video of the artificial culture lab online, together with visualisations of the data the experiments are generating, we hope that interested third parties will engage with and support the project in interpretation of its results. We are open to the engagement of others with very different perspectives on the project. What we hope is that through their observation they are able to discern patterns of behaviour that become traditions. Our exploratory engagement with school children suggests that they are enthusiastic about robots and enjoy watching them. Their comments about what they observe may provide us with prompts about emerging traditions. Although the childrens talk about robots is heavily influenced by their exposure to robots in films and robot toys, it is possible to hear within this discourse talk about what they observe the robots doing and how they differ from each other (Buckingham & Willett, 2006). By relating these comments in time to our record of what the robots were doing we are able to take a close look at the patterns they discern. The project team has considered whether the use of data mining techniques could suggest patterns that would be missed by people. At present we consider the use of data mining may be more powerful when guided by human observation. Given that we will have very detailed data tracking robot activity, it is likely we will be able to trace the roots of any robot traditions that emerge.

5.5.4 Robot Memes and Meme Tracking

Social learning, or robot-robot imitation, is clearly a key requirement of this project. Consider now what is being imitated and how we might track the evolution of robot memes through our robot society. We choose to limit ourselves to two modes of robot imitation, *movement* and *sound*. Thus, we propose a definition of a *robot meme* as follows:

> a contiguous sequence or package of movements, or sounds, copied from one robot to another, by imitation.

It is important to note here that we make no special distinction between movement memes or sound memes. Although sound is often a medium of communication (and hence language) in animals and humans we are careful to not assume that sound memes will form the basis for communication or proto-language in our robot society. Indeed we are decidedly neutral on the question of whether any emerging proto-cultural behaviours or traditions may or may not tend toward proto-language.

Fig. 5.46 illustrates robot-robot imitation of movement by graphically showing tracking data for both the original sequence of movements (describing a square) and

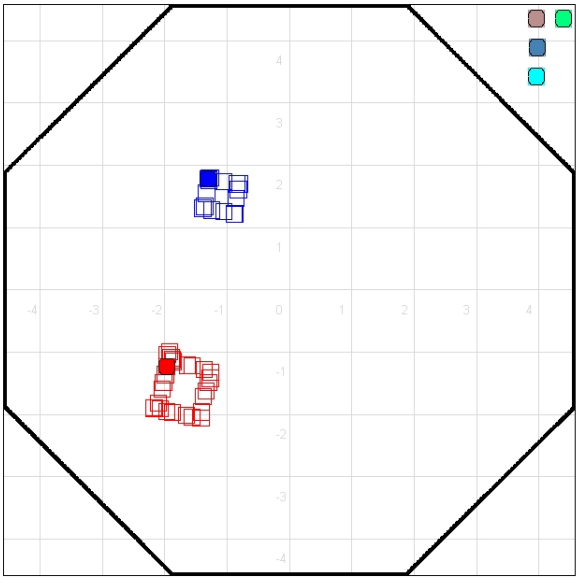


Fig. 5.46 Robot-robot movement imitation (captured by tracking two real robots). Robot A (upper, tracked in blue) has executed a sequence of movements to describe a square. Robot B (lower, tracked in red) has watched A then executed its imitated copy of A's movement sequence.

the imitated sequence. It is easy to see that qualitatively, the imitated movement sequence does indeed describe a square, but with some variation. Notably the imitated movement sequence shows both magnification and some rotation relative to the original; importantly these variations are emergent artefacts of the process of imitation and were not coded into the robot's imitation controller. Clearly we can measure the quantitative difference between the original sequence and its copy: the quality of imitation Q_i, where $Q_i = 1$ would represent perfect imitation.

Consider now not one imitation "event", as shown in Fig. 5.46, but a large number of such events taking place within our robot collective. Several different movement sequences (robot memes) might be active (i.e. being copied) at the same time. In order to begin to address the challenges and research questions discussed above we clearly need to be able to track the progress and evolution of these robot memes through the robot collective, over time. Fig. 5.47 suggests an approach for visually representing the evolution of robot memes within the robot collective. Each horizontal line represents an active meme, i.e. a meme that is being enacted and perhaps being copied; those enactions that are not imitated (possibly because no robot was watching at the time) are not shown. Imitation events (i.e. one robot successfully imitating another), such as the one shown in Fig. 5.46, are shown as blue diamonds. After a single imitation event we have a copy of the meme plus the original meme; the copy is clearly not identical to the original but, providing the quality of imitation

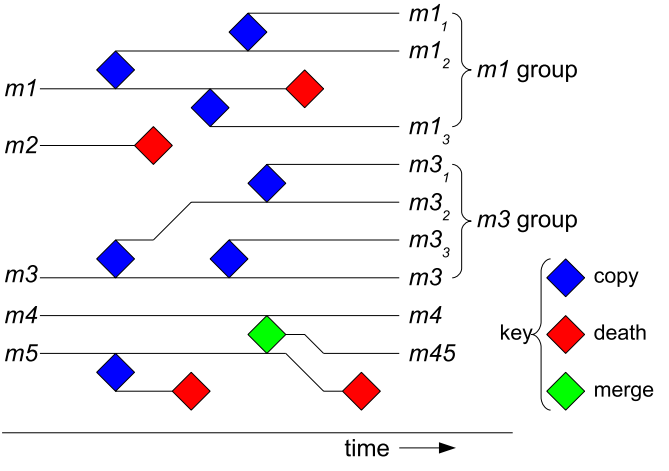


Fig. 5.47 A visualisation of a short time period during meme evolution within the robot collective. At the start of the period memes $m_1 ... m_5$ are present in the collective; horizontal lines represent the "life course" of the meme from left to right. Memes represent here either movement or sound sequences. (Note: this visualisation is not based on real data collected from experiments.)

is sufficiently high, the copy will be both quantitatively and qualitatively similar to the original. Thus we see in Fig. 5.47 memes $m1$ and $m3$ give rise to groups of related memes $(m1_1, m1_2, m1_3)$ and $(m3, m3_1, m3_2, m3_3)$, respectively where, for instance $m1_1$ is a copy of $m1$ (characterised by a given value of Q_i), and $m1_3$ is a copy of $m1_1$, i.e. a second generation copy.

Also shown in Fig. 5.47 are *meme death* events, as red diamonds. Meme death could, for instance, occur when the robot or robots which have an internal representation of that meme die. This is not so far fetched when we consider that in order for genetic evolution of robot controllers to take place (in parallel with memetic evolution), then robots will need to (genetically) evolve through successive generations. Thus it is necessary that the real physical robots in the artificial culture lab will have a time limited "life", after which the same physical robot is rebooted with an evolved controller (it becomes, in effect, a descendent). To retain biological plausibility a robot's imemes[17] will die with that robot. A second and equally plausible reason for meme death is that memes arguably should either have direct utility, or be directly associated with robot behaviours with utility; if a learned meme has no utility or value then the robot will not enact the meme, it therefore cannot be copied and will die when the robot comes to the end of its "life" (although we need to be cautious here as in observing robot culture we may have difficulty discerning utility). A third reason for meme death is that a robot may have sufficient memory only for a fixed number of imemes, so that older or low value memes are "forgotten"

[17] Here we use the word imeme as shorthand for "internal representation of a meme".

when new memes are learned. In Fig. 5.47 meme $m1$ dies, but not until after it has been copied three times; in contrast meme $m2$ dies before it is copied (perhaps for entirely contingent reasons). A third event is shown in Fig. 5.47, *meme merge*, as a green diamond. This represents the possibility that a robot may merge two memes into a third, perhaps because two robots are in its field of view, or within earshot, and are mistaken as a single robot.

This consideration of robot meme flow and meme evolution through a robot collective raises many interesting research questions. For instance, when is a copy sufficiently different to the original sequence (perhaps after several successive generations of imitation) that it can no longer be regarded as part of the same meme group, but a new meme in its own right? What are the upper and lower bounds of imitation fidelity (Q_i) outside which meme evolution breaks down either because there is no variation, or too much variation?

Finally we should note that the visualisation of meme evolution in Fig. 5.47 can readily also represent meme evolution within a system of gene-meme co-evolution simply by annotating imitation (and merge) events with the controller generation of the imitating robot for that event.

5.5.5 *Concluding Remarks*

At the time of writing it would be premature to draw any general conclusions with respect to either the overall aims of this project, or the subsidiary research questions generated by the project. However, we can be confident that within the context of multi-robot systems, especially those which utilise mechanisms of symbiosis and evolution, there is very considerable value in the development of techniques for robot-robot social learning. Whether those techniques will lead to the emergence of artificial culture in collective robot systems remains, for the time being, an open question[18].

Finally, let us briefly consider the work presented in this chapter in the context of multi-robot artificial organisms, the subject of this volume. The agents which collectively comprise the "robot society" are, in this chapter, single robots (e-pucks), and social learning (imitation) takes place between these single robots. However, we could choose a different level of abstraction for a society of robots so that the agents are instead multi-robot organisms, rather than single robots. In this case social learning (imitation) would need to take place between multi-robot organisms and all of the definitions (i.e. of robot memes) and arguments presented in this chapter apply equally. This raises the intriguing possibility of artificial culture emerging in a society of multi-robot organisms.

[18] The Artificial Culture project is funded by the UK Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/E062083/1. The authors gratefully acknowledge project co-investigators Alistair Sutcliffe, James Bown, Jenny Tennant Jackson and Robin Durie. Also gratefully acknowledged is the contribution of Wenguo Liu, BRL, for the design of the e-puck Linux extension board shown in Fig. 5.45(b).

Final Conclusions

Artificial multicellularity is a huge research field and this book only slightly touches several first conceptual issues. It must be developed and implemented further within a large interdisciplinary framework. Pursuing it must be seen as an essential conceptual and technological breakthrough for achieving extended reliability, advanced adaptivity and artificial evolution. It is a main enabler of sustainable development for the next generation of collective robotics and adaptive autonomous systems.

As presented in this book, artificial organisms reflect many similarities with biological organisms. This, in turn, opens a way for many great ideas, like artificial sexuality or inspiration from "natural chemistry", which may seen speculative on this moment, but are very promising in term of their biological analogies. Since research projects should be focused in concrete targeted goals, many of these ideas will be not followed further until their implementation into real systems. Therefore, one of the goals of this book is to provide to the readers possible earliest look at envisaged concepts and approaches, before Occams Razor will cut currently-not-feasible-ideas.

This book was not intended to give final answers, it is more intended to look for questions: what are plausible mechanisms of genetic deceases, does have the self-concept a leading role in a long-term unbound evolution, are information principles more relevant than energetic principles, unbounded artificial evolution vs. human-designed self-organization. The list of these question is long, we achieved at least one of our goals, when the reader starts to think about the same questions.

References

Abbeel, P.: Apprenticeship learning and reinforcement learning with application to robotic control. Ph.D. thesis, Stanford University, Dept. of Computer Science (2008)

Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Brodley, C.E. (ed.) Proc. of the Twenty-first International Conference on Machine Learning (ICML 2004). ACM International Conference Proceeding Series, vol. 69. ACM, Banff (2004)

Aboutalib, S., Veloso, M.: Towards using multiple cues for robust object recognition. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) Proc. of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), pp. 1152–1159. IFAAMAS, Honolulu (2007)

Achlioptas, D.: Random Matrices in Data Analysis. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 1–7. Springer, Heidelberg (2004)

Agassounon, W.: Modeling Artifical, Mobile Swarm Systems. Ph.D. Thesis, California Institute of Technology (2003)

Agassounon, W., Martinoli, A.: A Macroscopic Model of an Aggregation Experiment using Embodied Agents in Groups of Time-Varying Sizes. In: Proc. of the IEEE International Conference on Systems, Man and Cybernetics, SMC 2002 (2002)

Aghajan, H., Cavallaro, A.: Multi-Camera Networks: Principles and Applications. Academic Press, London (2009)

Alba, E., Tomassini, M.: Parallelism and Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 6(5), 443–462 (2002)

Alberts, B., Johnson, A., Walter, P., Lewis, J., Raff, M., Roberts, K., Orme, N.: Molecular Biology of the Cell. Taylor & Francis, Abington (2008)

Alippi, C., Vanini, G.: A RSSI-based and calibrated centralized localization technique for Wireless Sensor Networks. In: Proc. of the 4th IEEE Conference on Pervasive Computing and Communications Workshops (PerComW 2006), pp. 301–305. IEEE Computer Society, Pisa (2006)

Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Solving the correspondence problem in robotic imitation across embodiments: synchrony, perception and culture in artefacts. In: Nehaniv, C., Dautenhahn, K. (eds.) Imitation and Social Learning in Robots, Humans and Animals, pp. 249–273. Cambridge University Press, Cambridge (2007)

Analog Devices. Blackfin Embedded Processor ADSP-BF534/ADSP-BF536/ADSP-BF537. Rev. g 2009s edn. (2009)

Andersen, T., Newman, R., Otter, T.: Shape Homeostasis in Virtual Embryos. Artificial life 15(2), 161–183 (2009)

Anderson, B.D.O.: Failures of adaptive control theory and their resolution. Communications in information and systems 5(1), 1–20 (2005)

Anderson, B.D.O., Bitmead, R.R., Johnson, C.R.J., Kokotovic, P.V., Kosut, R.L., Mareels, I.M.Y., Praly, L., Riedle, B.D.: Stability of Adaptive Systems: Passivity and Averaging Analysis. MIT Press, Berlin (1986)

Ansari, J., Riihijarvi, J., Mahonen, P.: Combining Particle Filtering with Cricket System for Indoor Localization and Tracking Services. In: Proc. of the 18th International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC 2007), pp. 1–5 (2007)

Arkin, R.C., Mackenzie, D.C.: Planning to Behave: A Hybrid Deliberative/Reactive Robot Control Architecture for Mobile Manipulation. In: Proc. of the Fifth International Symposium on Robotics and Manufacturing, vol. 5, pp. 5–12 (1994)

Arnold, V.I.: Dynamical systems III. Springer, Heidelberg (1988)

ATMEL. 8-bit AVR Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash ATmega640/V ATmega1280/V. Rev. 2549l-avr-08/07 edn. (2007)

ATMEL. 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash ATmega128 ATmega128L. Rev. 2467s-avr-07/09 edn. (2009a)

ATMEL. 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash ATmega16 ATmega16L. Rev. 2466ss-avr-05/10 edn. (2009b)

ATMEL. 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash ATmega8 ATmega8L. Rev. 2486v-avr-05/09 edn. (2009c)

Avila-García, O., Cañamero, L.: Hormonal Modulation of Perception in Motivation-Based Action Selection Architectures. In: Cañamero, L. (ed.) Agents that Want and Like: Motivational and Emotional Roots of Cognition and Action, Symposium of the AISB 2005 Convention, pp. 700–712. University of Hertfordshire (2005)

Avrutin, V.: Bifurcation Structures within Robust Chaos, Habilitation (Dr. of Science). Faculty of Computer Science, Electrical Engineering and Information Technology, University Stuttgart (2009)

Avstreih, A.K.: The Emerging Self: Psychoanalytic Concepts of Self Development and Their Implications for Dance Therapy. American Journal of Dance Therapy 4(2), 21–32 (1981)

Baars, B.J.: A cognitive theory of consciousness. Cambridge University Press, Cambridge (1988)

Babloyantz, A., Hiernaux, J.: Models for Positional Information and Positional Differentiation. National Academy of Sciences 71(4), 1530–1533 (1974)

Bäck, T., Hammel, U., Schwefel, H.-P.: Evolutionary Computation: Comments on the History and Current State. Transactions on Evolutionary Computation 1(1), 3–17 (1997)

Baele, G., Bredeche, N., Haasdijk, E., Maere, S., Michiels, N., Van de Peer, Y., Schmickl, T., Schwarzer, C., Thenius, R.: Open-ended On-board Evolutionary Robotics for Robot Swarms. In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC 2009). IEEE Press, Trondheim (2009)

Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-based User Location and Tracking System. In: Proc. of the 19th IEEE Conference on Computer Communications (IEEE INFOCOM), pp. 775–784 (2000)

Bain, M., Sammut, C.: A framework for behavioural cloning. In: Furukawa, K., Michie, D., Muggleton, S. (eds.) Machine Intelligence, vol. 15, pp. 103–129. Oxford University Press, Oxford (1995)

Baldwin, J.M.: A new factor in evolution. American Naturalist 30(354), 441–451 (1896)
Ball, R.: A Treatise on the Theory of Screws. Cambridge University Press, Cambridge (1900)
Balzani, V., Venturi, M., Credi, A.: Molecular Devices and Machines A Journey into the Nanoworld. Wiley-VCH, Weinheim (2003)
Barrera, A., Kortenkamp, D., Bonasso, R.P., Murphy, R.: Artificial Intelligence and Mobile Robots. Automatica 41(4), 737–738 (2005)
Barreto, G.D.A., Araújo, A.F.R., Ritter, H.: Self-Organizing Feature Maps for Modeling and Control of Robotic Manipulators. Journal of Intelligent and Robotic Systems 36(4), 407–450 (2003)
Basanta, D., Miodownik, M., Baum, B.: The Evolution of Robust Development and Homeostasis in Artificial Organisms. PLoS Computational Biology 4(3) (2008)
Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (2008)
Beer, R.D.: Intelligence as Adaptive Behaviour: An Experiment in Computational Neuroethology. Academic Press, London (1990)
Bender, J.: Impulse-based dynamic simulation in linear time. Computer Animation and Virtual Worlds 18(4-5), 225–233 (2007)
Beni, G.: From Swarm Intelligence to Swarm Robotics. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 1–9. Springer, Heidelberg (2005)
Benini, L., de Micheli, G.: Dynamic Power Management: Design Techniques and CAD Tools. Kluwer Academic Publishers, Norwell (1998)
Bennetto, P.: Microbes come to power. New Scientist 114, 36–39 (1987)
Berek, C., Ziegner, M.: The Maturation of the Immune Response. Immunology Today 14, 200–402 (1993)
Beyer, H.-G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering 186(2-4), 239–267 (2000)
Bickel, S., Sawade, C., Scheffer, T.: Transfer Learning by Distribution Matching for Targeted Advertising. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Proc. of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS 2009), pp. 145–152. MIT Press, Vancouver (2009)
Birman, J.S.: Braids, knots and contact structures (2004)
Bishop, C.M.: Novelty Detection and neural network validation. IEE Proceedings - Vision, Image, and Signal Processing 141(4) (1994)
Bjerknes, J.D.: Scaling and Fault Tolerance in Self-organised Swarms of Mobile Robots. Ph.D. thesis, University of the West of England, Bristol (2010)
Bjerknes, J.D., Winfield, A.F.T., Melhuish, C.R.: An Analysis of Emergent Taxis in a Wireless Connected Swarm of Mobile Robots. In: Proc. of the IEEE Swarm Intelligence Symposium (SIS 2007), pp. 45–52 (2007)
Bjorken, J., Drell, S.: Relativistic Quantum Fields. McGraw-Hill, New York (1965)
Blackmore, S.: The Meme Machine. Oxford University Press, Oxford (1999)
Boletis, A., Driesen, W., Breguet, J.-M., Brunete, A.: Solar Cell Powering with Integrated Global Positioning System for mm3 Size Robots. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), pp. 5528–5533. IEEE, Beijing (2006)
Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, Oxford (1999)

Bonarini, A., Lazaric, A., Restelli, M., Vitali, P.: Self-Development Framework for Reinforcement Learning Agents. In: Proc. of the 5th International Conference on Development and Learning, ICDL 2006 (2006)

Bongard, J., Zykov, V., Lipson, H.: Resilient Machines Through Continuous Self-Modeling. Science 314(5802), 1118–1121 (2006)

Bongard, J.C., Paul, C.: Making Evolution an Offer It Can't Refuse: Morphology and the Extradimensional Bypass. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 401–412. Springer, Heidelberg (2001)

Bongard, J.C., Pfeifer, R.: Evolving complete agents using artificial ontogeny. Morphofunctional Machines: The New Species (Designing Embodied Intelligence), 237–258 (2003)

Bordignon, M., Stoy, K., Christensen, D.J., Schultz, U.P.: Towards Interactive Programming of Modular Robots. In: Proc. of the IROS 2008 Workshop on Self-Reconfigurable Robots, Systems and Applications (2008)

Borenstein, J., Everett, B., Feng, L.: Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Wellesley (1996)

Bound, K.E., Tollin, G.: Phototactic Response of *Euglena gracilis* to Polarized Light. Nature 216, 1042–1044 (1967)

Braitenberg, V.: Vehicles: experiments in synthetic psychology. MIT Press, Cambridge (1984)

Bratman, M.E.: Intention, Plans, and Practical Reason. Harvard University Press, Cambridge (1987)

Breazeal, C.: Emotion and sociable humanoid robots. International Journal of Human-Computer Studies 59(1-2), 119–155 (2003)

Breazeal, C., Scassellati, B.: A context-dependent attention system for a social robot. In: Dean, T. (ed.) Proc. of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 1146–1153 (1999)

Bredeche, N., Haasdijk, E., Eiben, A.E.: On-line, On-board Evolution of Robot Controllers. In: Proc. of the 9th international conference on Artificial Evolution (Evolution Artificielle 2009). LNCS. Springer, Heidelberg (2009)

Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)

Brener, N., Ben Amar, F., Bidaud, P.: Designing Modular Lattice Systems with Chiral Space Groups. Int. J. Rob. Res. 27(3-4), 279–297 (2008)

Britannica, Encyclopedia, Homeostasis (2009)

Brockett, R.W.: Mathematical Theory of Networks and Systems. Springer, Heidelberg (1984)

Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. Journal of Robotics and Automation 2(1), 14–23 (1986)

Bryson, J.J. (ed.): Special Issue: Mechanisms of Action Selection. vol. 15 (2007)

Buckingham, D., Willett, R.: Digital Generations: Children, Young People and New Media. Lawrence Erlbaum, New Jersey (2006)

Bull, L., Studley, M., Bagnall, A., Whitley, I.: Learning Classifier System Ensembles With Rule-Sharing. IEEE Transactions on Evolutionary Computation 11(4), 496–502 (2007)

Burgard, W., Fox, D., Hennig, D., Schmidt, T.: Estimating the absolute position of a mobile robot using position probability grids (1996)

Burges, C.J.C., Ragno, R., Le, Q.V.: Learning to Rank with Nonsmooth Cost Functions. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems, vol. 19, pp. 193–200. MIT Press, Cambridge (2007)

Burkard, R., Dell'Amico, M., Martello, S. (eds.): Assignment Problems. Society for Industrial and Applied Mathematics (2009)

Burnet, F.M.: The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge (1959)

Burtsev, M.S.: Model of evolutionary emergence of goal-directed adaptive behavior. 2. Investigation of development of hierarchy of goals (in Russian). Preprint of M.V. Keldysh Institute of Applied Mathematics (2002)

Burtsev, M.S., Turchin, P.V.: Evolution of cooperative strategies from first principles. Nature 440(7087), 1041–1044 (2006)

Bushev, M.: Synergetics: chaos, order, self-organization. World Scientific Publisher, Singapore (1994)

Butz, M.: Anticipatory Learning Classifier Systems. In: Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)

Cabrera, B.J., Rust, M.K.: Caste differences in feeding and trophallaxis in the western drywood termite, Incisitermes minor (Hagen) (Isoptera, Kalotermitidae). Insectes Sociaux 46(3), 244–249 (1999)

Calabretta, R., Galbiati, R., Nolfi, S., Parisi, D.: Two is better than one: a diploid genotype for neural networks. Neural Processing Letters 4(3), 149–155 (1996)

Calabretta, R., Nolfi, S., Parisi, D.: Investigating the role of diploidy in simulated populations of evolving individuals. In: Husbands, P., Harvey, I. (eds.) Proc. of the Fourth European Conference on Artificial Life. The MIT Press, Brighton (1997)

Calabretta, R., Nolfi, S., Parisi, D., Wagner, G.P.: A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro-and cognitive science. In: Adami, C., Belew, R.K., Kitano, H., Taylor, C. (eds.) Proc. of the sixth international conference on Artificial life, pp. 275–284. The MIT Press, Cambridge (1998)

Calabretta, R., Nolfi, S., Parisi, D., Wager, G.P.: Duplication of Modules Facilitates the Evolution of Functional Specialization. Artificial Life 6, 69–84 (2000)

Calinon, S., Billard, A.: Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM. In: Raedt, L.D., Wrobel, S. (eds.) Proc. of the Twenty-Second International Conference on Machine Learning (ICML 2005). ACM International Conference Proceeding Series, vol. 119, pp. 105–112. ACM, Bonn (2005)

Camazine, S.: The regulation of pollen foraging by honey bees: how foragers assess the colony's need for pollen. Behavioral Ecology and Sociobiology 32(4), 265–272 (1993)

Camazine, S., Deneubourg, J.L., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2001)

Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2003)

Carlson, J., Murphy, R.R.: Reliability Analysis of Mobile Robots. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2003), pp. 274–281. IEEE, Los Alamitos (2003)

Carpenter, G.A., Grossberg, S.: Adaptive Resonance Theory. Boston University, Center for Adaptive Systems and Dept. of Cognitive and Neural Systems (1994)

Carroll, S.B.: Endless Forms Most Beautiful: The New Science of Evo Devo. W. W. Norton, New York (2006)

Carter, J., Saunders, V.: Virology: Principles and Applications. Wiley, Chichester (1997)

Castano, A., Will, P.: Representing and Discovering the Configuration of CONRO Robots. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA 2001), vol. 4, pp. 3503–3509. IEEE, Los Alamitos (2001)

Castano, A., Shen, W.-M., Will, P.: CONRO: Towards Deployable Robots with Inter-Robots Metamorphic Capabilities. Journal of Autonomous Robots 8, 309–324 (2000)

Cayzer, S., Smith, J., Marshal, J., Kovacs, T.: What have gene libraries done for AIS. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 86–99. Springer, Heidelberg (2005)

Cesa-Bianchi, N., Lugosi, G.: Potential-based algorithms in on-line prediction and game theory. Machine Learning 51, 239–261 (2003)

Chae, H., Yu, W., Lee, J., Cho, Y.: Robot Localization Sensor for Development of Wireless Location Sensing Network (2006)

Chen, I.-M., Yang, G.: Automatic Model Generation for Modular Reconfigurable Robot Dynamics. ASME Journal of Dynamic Systems, Measurement, and Control 120, 346–352 (1999)

Chen, I.-M.: Theory and Applications of Modular Reconfigurable Robotics Systems. Ph.D. thesis, Calfornia Institute of Technology, CA (1994)

Cheng, B.H.C., Giese, H., Inverardi, P., Magee, J., de Lemos, R.: Software Engineering for Self-Adaptive Systems. Dagstuhl Seminar 08031 (2008)

Chevallier, S., Ijspeert, A.J., Ryczko, D., Nagy, F., Cabelguen, J.-M.: Organisation of the spinal central pattern generators for locomotion in the salamander: biology and modelling. Brain Research Reviews 57(1), 147–161 (2008)

Chiang, C.-J., Chirikjian, G.S.: Modular Robot Motion Planning Using Similarity Metrics. Auton. Robots 10(1), 91–106 (2001)

Chiel, H.J., Beer, R.D., Quinn, R.D., Espenschied, K.S.: Robustness of a distributed neural network controller for locomotion in a hexapod robot. IEEE Transactions on Robotics and Automation 8(3), 293–303 (1992)

Chin, W., Ott, E., Nusse, H.E., Grebogi, C.: Grazing Bifurcation in Impact Oscillators. Physical Review E 50(6), 4427–4444 (1994)

Christensen, A., O'Grady, R., Dorigo, M.: From Fireflies to Fault Tolerant Swarms of Robots. IEEE Transactions on Evolutionary Computation 13(4), 754–766 (2009)

Christensen, D.J.: Evolution of shape-changing and self-repairing control for the ATRON self-reconfigurable robot. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2006), pp. 2539–2545 (2006)

Clack, J.A.: Getting a leg up on land. Scientific American, 293(6), 100–107 (2005); ISI Document Delivery No.: 990NJ Times Cited: 1 Cited Reference Count: 2 SCI AMERICAN INC

Cliff, D.: Biologically-Inspired Computing Approaches To Cognitive Systems: a partial tour of the literature. Hewlett-Packard Company (2003)

Cohen, I.R.: Tending Adam's Garden: Evolving the Cognitive Immune Self. Academic Press, London (2000)

Colegrave, N.: Sex releases the speed limit on evolution. Nature, 420(6916), 664–666 (2002); ISI Document Delivery No.: 624GK Times Cited: 45 Cited Reference Count: 15 NATURE PUBLISHING GROUP

Conrad, M.: Adaptability: The Significance of Variability from Molecule to Ecosystem. Plenum Press, New York (1999)

Constantinescu, C., Kornienko, S., Kornienko, O., Heinkel, U.: An agent-based approach to support the scalability of change propagation. In: Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (ISCA 2004), pp. 157–164. ISCA, San-Francisco (2004)

Correll, N., Martinoli, A.: Modeling and Analysis of Beaconless and Beacon-Based Policies for a Swarm-Intelligent Inspection System. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 2477–2482. IEEE, Barcelona (2005)

Correll, N., Martinoli, A.: Modeling and Optimization of a Swarm-Intelligent Inspection System. In: Proc. of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS 2006). Distributed Autonomous Robotic Systems. Springer, Heidelberg (2006)

Correll, N., Martinoli, A.: Towards Optimal Control of Self-Organized Robotic Inspection Systems. In: Proc. of the 8th International IFAC Symposium on Robot Control (SYROCO 2006) (2006b)

Correll, N., Martinoli, A.: Modeling Self-Organized Aggregation in a Swarm of Miniature Robots. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2007) Workshop on Collective Behaviors inspired by Biological and Biochemical Systems (2007)

Cox, I.J.: Blanche–an experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transactions on Robotics and Automation 7, 193–204 (1991)

Crabbe, F.L.: Compromise Strategies for Action Selection. Philosophical Transactions of the Royal Society. B. 362, 1559–1571 (2007)

Crick, F.: Diffusion in embryogenesis. Nature 225, 420–422 (1970)

Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)

Cullen, K.: Sensory signals during active versus passive movement. Current Opinion in Neurobiology 14(6), 698–706 (2004)

Dahmann, J.S.: Standards for Simulation: As Simple As Possible But Not Simpler The High Level Architecture For Simulation. SIMULATION 71(6), 378–387 (1998)

Dailey, K.W.: The FMEA Handbook. DW Publishing (2004)

D'Ambrosio, D.B., Stanley, K.O.: A novel generative encoding for exploiting neural network sensor and output geometry. In: Proc. of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 974–981 (2007)

Darnton, N.C., Turner, L., Rojevsky, S., Berg, H.C.: On Torque and Tumbling in Swimming *Escherichia coli*. Journal of Bacteriology 189(5), 1756–1764 (2007)

Darpa-Urban: Results of the Darpa Urban Challenge. Darpa Web site (2007), `http://www.darpa.mil/grandchallenge/index.asp`

Dasgupta, S.: Coarse sample complexity bounds for active learning. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems (NIPS 2005), pp. 235–242. MIT Press, Cambridge (2006)

Dautenhahn, K.: Getting to Know Each Other - Artificial Social Intelligence for Autonomous Robots. Robotics and Autonomous Systems 16, 333–356 (1995)

Davidson, J.K., Hunt, K.H.: Robots and Screw Theory: Applications of Kinematics and Statics to Robotics. Oxford University Press, Oxford (2004)

Davis, P.: Circulant matrices. John Wiley & Sons, Chichester (1979)

Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)

Dawkins, R.: The Blind Watchmaker. W. W. Norton & Company, Inc., New York (1986 - 1996)

Dayan, P., Abbott, L.F.: Theoretical Neuroscience. MIT Press, Cambridge (2001)

de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)

de Lemos, R., Timmis, J., Forrest, S., Ayara, M.: Immune-Inspired Adaptable Error Detection for Automated Teller Machines. IEEE Trans. on Systems, Man and Cybernetics Part C: Applications and Reviews 37(5), 873–886 (2007)

Delarboulas, P., Sebag, M.: Critères non supervisés pour contrôleurs robotiques embarqués. In: Proc. of the Reconnaissance des Formes et Intelligence Artificielle, RFIA 10 (to appear, 2010)

Denavit, J., Hartenberg, R.S.: A Kinematic Notation for Lower Pair Mechanisms Based on Matrices. Trans. ASME J. Applied Mechanics 22, 215–221 (1995)

Derbeko, P., El-Yaniv, R., Meir, R.: Variance Optimized Bagging. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 60–71. Springer, Heidelberg (2002)

Dictionary: Mosby's Dental. Homeostasis, cell (2008)

Doerr, B., Happ, E., Klein, C.: Crossover Can Provably be Useful in Evolutionary Computation. In: Ryan, C., Keijzer, M. (eds.) Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 539–546 (2008)

Domingo, E., Holland, J.J.: RNA Virus Mutations and Fitness for Survival. Annual Review of Microbiology 51, 151–178 (1997)

Dorigo, M., Şahin, E.: Guest editorial: Swarm robotics. Autonomous Robots 17(2-3), 111–113 (2004)

Dormann, D., Weijer, C., Siegert, F.: Twisted scroll waves organize Dictyostelium mucoroides slugs. Journal of Cell Science 110, 1831–1837 (1997)

Drogoul, A., Zucker, J.: Methodological Issues for Designing Multi-Agent Systems with Machine Learning Techniques: Capitalizing Experiences from the RoboCup Challenge. Tech. rept. Laboratoire d'Informatique de Paris 6 (1998)

Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, Hoboken (2001)

Durfee, E.H.: Distributed Problem Solving and Planning. In: Weiss, G. (ed.) Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence, pp. 121–164. MIT Press, Cambridge (1999)

Durrant-Whyte, H., Bailey, T.: Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. Robotics and Automation Magazine 13, 99–110 (2006)

Eckert, J., Dressler, F., German, R.: An Indoor Localization Framework for Four-rotor Flying Robots Using Low-power Sensor Nodes. Tech. rept. 0209. University of Erlangen, Dept. of Computer Science 7 (2009)

Egardt, B.: Stability of Adaptive Controllers. LNCIS, vol. 20. Springer, Berlin (1979)

Eiben, A.E., Griffioen, A.R., Haasdijk, E.: Population-based Adaptive Systems: concepts, issues, and the platform NEW TIES. In: Proc. of the European Conference on Complex Systems, ECCS 2007 (2007)

Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Natural Computing Series. Springer, Heidelberg (2003)

Eichenbaum, H.: A cortical–hippocampal system for declarative memory. Nature Reviews Neuroscience 1(1), 41–50 (2000)

Eigen, M.: Molekulare Selbstorganisation und Evolution (Selforganization of matter and the evolution of biological macromolecules). Naturwissenschaften 58(10), 465–523 (1971)

Eigen, M., Schuster, P.: The Hypercycle: A Principle of Natural Self-Organization. Springer, Berlin (1979)

Elgebaly, M., Sachdev, M.: Efficient Adaptive Voltage Scaling System Through On-Chip Critical Path Emulation. In: Joshi, R.V., Choi, K., Tiwari, V., Roy, K. (eds.) Proc. of the International Symposium on Low Power Electronics and Design (ISLPED 2004), pp. 375–380. ACM, Newport Beach (2004)

Elman, J.L.: Finding structure in time. Cognitive Science 14(2), 179–211 (1990)

Elsayed, E.A.: Reliability Engineering. Addison Wesley Longman, Amsterdam (1996)

Ephraim, Y., Merhav, N.: Hidden Markov processes. IEEE Transactions on Information Theory 48(6), 1518–1569 (2002)

Espinace, P., Soto, A., Torriti, M.: Real-Time Robot Localization in Indoor Environments Using Structural Information (2008)

Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, Heidelberg (2008)

Ferber, J.: Multi-Agent Systems. Addison-Wesley, Harlow (1999)

Ficici, S.G., Watson, R.A., Pollack, J.B.: Embodied Evolution: A Response to Challenges in Evolutionary Robotics. In: Demiris, J., Wyatt, J.L. (eds.) EWLR 1999. LNCS (LNAI), vol. 1812, pp. 14–22. Springer, Heidelberg (2000)

Flener, P., Frisch, A.M., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., Walsh, T.: Breaking Row and Column Symmetries in Matrix Models. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 462–476. Springer, Heidelberg (2002)

Floreano, D., Mattiussi, C.: Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies. MIT Press, Cambridge (2008)

Floreano, D., Mondada, F.: Evolution of homing navigation in a real mobile robot. IEEE Transactions on Systems, Man, and Cybernetics 26, 396–407 (1994)

Floreano, D., Husbands, P., Nolfi, S.: Evolutionary Robotics. In: Handbook of Robotics. Springer, Berlin (2008a)

Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence 1, 47–62 (2008b)

Forsman, A., Hagman, M.: Calling is an honest indicator of paternal genetic quality in poison frogs. Evolution 60(10), 2148–2157 (2006)

Fox, M., Ghallab, M., Infantes, G., Long, D.: Robot introspection through learned hidden Markov models. Artif. Intell. 170(2), 59–113 (2006)

Franklin, S., Ferkin, M.H.: An ontology for comparative cognition: A functional approach. Comparative Cognition & Behavior Reviews 1, 36–52 (2006)

Freund, Y., Shapire, R.E.: Experiments with a new boosting algorithm. In: Saitta, L. (ed.) Proc. of the 13th International Conference on Machine Learning (ICML 1996), pp. 148–156. Morgan Kaufmann, Bari (1996)

Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4(6), 933–969 (2003)

Fukuda, T., Ueyama, T.: Cellular robotics and micro robotic systems. World Scientific Publishing Co. Pte. Ltd, Singapore (1994)

Fukuda, T., Liu, P., Kantola, K.: Nanoscale cutting, bending and welding in a nanoassembly. In: Proc. of the 2nd IEEE International Conference on Nano/Micro Engineered and Molecular Systems (IEEE NEMS 2007), pp. 548–551. IEEE, Bangkok (2007)

Fullford, D.: Distributed interactive simulation: it's past, present, and future. In: Charnes, J., Morrice, D.J., Brunner, D.T., Swain, J.J. (eds.) Proc. of the Winter Simulation Conference (WSC 1996), pp. 179–185. IEEE, Coronado (1996)

Funda, J., Pual, R.P.: A Computational Analysis of Screw Transformation in Robotics. IEEE Transactions on Robotics and Automation 6(3), 348–356 (1990)

Gagné, C., Sebag, M., Schoenauer, M., Tomassini, M.: Ensemble Learning for Free with Evolutionary Algorithms? In: Thierens, D. (ed.) Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 1782–1789. ACM Press, London (2007)

Galan, R., et al.: A systematic approach for product families formation in Reconfigurable Manufacturing Systems. Robot. Comput. Integr. Manuf. 23(5), 489–502 (2007)

Galstyan, A., Lerman, K.: Analysis of a Stochastic Model of Adaptive Task Allocation in Robots. In: Brueckner, S., Serugendo, G.D.M., Karageorgos, A., Nagpal, R. (eds.) ESOA 2005. LNCS (LNAI), vol. 3464, pp. 167–179. Springer, Heidelberg (2005)

Gauci, J., Stanley, K.O.: A case study on the critical role of geometric regularity in machine learning. In: Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008). AAAI Press, Menlo Park (2008)

Gerkey, B.P., Vaughan, R.T., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: Proc. of the 11th International Conference on Advanced Robotics, pp. 317–323 (2003)

Germain, R.N., Stefanova, I.: The dynamics of T cell receptor signalling: complex orchestration and the key roles of tempo and cooperation. A. Rev. Imm. 17, 467–522 (1999)

Ghrist, R.W., Holmes, P.J., Sullivan, M.C.: Knots and Links in Three-Dimensional Flows. Springer, Heidelberg (1997)

Gienger, M., Toussaint, M., Goerick, C.: Task maps in humanoid robot manipulation. In: Proc. of the International Conference on Intelligent Robots and Systems (IROS 2008), pp. 2758–2764. IEEE, Nice (2008)

Gierer, A., Meinhardt, H.: A Theory of Biological Pattern Formation. Kybernetik 12, 30–39 (1972)

Gilpin, K., Kotay, K., Rus, D., Vasilescu, I.: Miche: Modular Shape Formation by Self-Disassembly. Int. J. Rob. Res. 27(3-4), 345–372 (2008)

Gobet, F., Simon, H.A.: Templates in Chess Memory: A Mechanism for Recalling Several Boards. Cognirtive Psychology 31, 1–40 (1996)

Godzik, N., Schoenauer, M., Sebag, M.: Evolving Symbolic Controllers. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 638–650. Springer, Heidelberg (2003)

Godzik, N., Schoenauer, M., Sebag, M.: Robustness in the long run: Auto-teaching vs. Anticipation in Evolutionary Robotics. In: Yao, X., Burke, E., Lozano, J.A., Smith, J., Merelo-Guervos, J., Bullinaria, J.A., Rowe, J., Tino, P., Kaban, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 932–941. Springer, Heidelberg (2004)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)

Gomperts, B.D., Kramer, I.M., Tatham, P.E.R.: Signal transduction. Academic Press, London (2002)

González, F.M.: The Coevolution of Robot Behavior and Central Action Selection. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007. LNCS, vol. 4528, pp. 439–448. Springer, Heidelberg (2007)

González, F.M., Reyes, J.S., Ríos-Figueroa, H.V.: Integration of Evolution with a Robot Action Selection Model. In: Gelbukh, A.F., García, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. 4293, pp. 1160–1170. Springer, Heidelberg (2006)

Granlund, G.H.: Organization of Architectures for Cognitive Vision Systems. In: Christensen, H.I., Nagel, H.-H. (eds.) Cognitive Vision Systems. LNCS, vol. 3948, pp. 37–55. Springer, Heidelberg (2006)

Green, M.A., Emery, K., Hishikawa, Y., Warta, W.: Solar cell efficiency tables (Version 34). Progress in Photovoltaics: Research and Applications 17(5), 320–326 (2009)

Grossberg, S.: Competitive learning: From interactive activation to adaptive resonance. Cognitive Science 11(1), 23–63 (1987)

Guckenheimer, J., Holmes, P.J.: Nonlinear oscillations, dynamical systems, and bifurcations of vector fields. Springer, Berlin (1983)

Gurvis, J., Calarco, A.: Adaptability: Responding Effectively to Change. Pfeiffer (2007)

Gustafsson, F., Gunnarsson, F.: Positioning using timedifference of arrival measurements (2003)
Häder, D., Colombetti, G., Lenci, F., Quaglia, M.: Phototaxis in the flagellates, *Euglena gracilis* and *Ochromonas danica*. Archives of Microbiology 130(1), 78–82 (1981)
Haken, H.: Laser Theory, Handbuch der Physik XXV/2c. Springer, Heidelberg (1970)
Haken, H.: Synergetics: An introduction. Springer, Heidelberg (1977)
Haken, H.: Synergetics: Introduction and Advanced Topics. Springer, Heidelberg (1983)
Haken, H.: Light 2: Laser Light Dynamics. Elsevier, Amsterdam (1985)
Haken, H.: Information and Self-Organisation. Springer, Heidelberg (1988)
Haken, H.: Synergetics: Introduction and Advanced Topics. Springer, Heidelberg (2004)
Haken, H., Wolf, H.: Molekülphysik und Quantenchemie. Springer, Heidelberg (1998)
Hamann, H.: Pattern Formation as a Transient Phenomenon in the Nonlinear Dynamics of a Multi-Agent System. In: Troch, I., Breitenecker, F. (eds.) Proc. of the 6th Vienna International Conference on Mathematical Modelling, MATHMOD 2009 (2009)
Hamilton, W.D.: The genetical evolution of social behaviour I&II. Journal of Theoretical Biology 7(1), 1–52 (1964)
Hampton, A.N., Adami, C.: Evolution of Robust Developmental Neural Networks. ArXiv Nonlinear Sciences e-prints (May 2004)
Hand, W.G., Haupt, W.: Flagellar Activity of the Colony Members of Volvox aureus Ehrbg. during Light Stimulation. Journal of Eukaryotic Microbiology 18(3), 361–364 (1971)
Harada, K., Susilo, E., Menciassi, A., Dario, P.: Wireless reconfigurable modules for robotic endoluminal surgery. In: Proc. of IEEE/ASME International Conference on Robotics and Automation, ICRA 2009 (2009)
Hart, E., Timmis, J.: Application Areas of AIS: The Past, The Present and the Future. Applied Soft Computing 8(1), 191–201 (2008)
Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Proc. of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp. 59–68 (1999)
Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. Wireless Networks 8, 187–197 (2002)
Hartland, C., Bredeche, N., Sebag, M.: Memory-Enhanced Evolutionary Robotics: The Echo State Network Approach. In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC 2009), IEEE Press, Trondheim (2009)
Hastad, J.: Almost optimal lower bounds for small depth circuits. In: Proc. of the Eighteenth Annual ACM Symposium on Theory of Computing (STOC 1986), pp. 6–20. ACM Press, New York (1986)
Hayes, A.T., Martinoli, A., Goodman, R.M.: Comparing distributed exploration strategies with simulated and real autonomous robots. In: Parker, L.E., Bekey, G., Bahren, J. (eds.) Proc. of the 5th International Symposium on Distributed Autonomous Robotic System (DARS 2000), pp. 261–270. Springer, Heidelberg (2000)
Hexmoor, H., Horswill, I., Kortenkamp, D. (eds.): Special Issue: Software Architectures for Hardware Agents 9(2) (1997)
Hinton, G.E., Osindero, S., Teh, Y.-W.: A Fast Learning Algorithm for Deep Belief Nets. Neural Comp. 18(7), 1527–1554 (2006)
Hoffmann, G.W.: Co-selection in immune network theory and in AIDS pathogenesis. Immunology and Cell Biology 72, 338–346 (1994)
Hoffmann, H.: Perception through visuomotor anticipation in a mobile robot. Neural Networks 20(1), 22–33 (2007)
Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Boston (1992)

Hollerbach, J.M.: Optimum kinematic design for a seven degree of freedom manipulator. In: Hanafusa, H., Inoue, H. (eds.) Proc. of the Second International Symposium on Robotics Research (ISRR 1984), pp. 341–349. MIT Press, Cambridge (1984)

Holmes, S.J.: Phototaxis in Volvox. Biological Bulletin 4, 319–326 (1903)

Hopler, R., Mosterman, P.J.: Model integrated computing in robot control to synthesize real-time embedded code. In: Proc. of the IEEE International Conference on Control Applications (CCA 2001), pp. 767–772 (2001)

Hopler, R., Otter, M.: A versatile C++ toolbox for model based, real time control systems of robotic manipulators. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), vol. 4, pp. 2208–2214 (2001)

Hornby, G.S., Pollack, J.B.: Evolving L-Systems To Generate Virtual Creatures. Computers and Graphics 25, 1041–1048 (2001)

Hou, F., Shen, W.-M.: Hormone-inspired Adaptive Distributed Synchronization of Reconfigurable Robots. In: Arai, T., Pfeifer, R., Balch, T.R., Yokoi, H. (eds.) Proc. of the 9th International Conference on Intelligent Autonomous Systems (IAS-9), IOS Press, Tokyo (2006a)

Hou, F., Shen, W.-M.: Mathematical Foundation for Hormone-Inspired Control for Self-Reconfigurable Robotic Systems. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2006), pp. 1477–1482. IEEE, Orlando (2006b)

Howson, C., Urbach, P.: Scientific Reasoning: The Bayesian Approach, 2nd edn. Open Court Pub. Co., Chicago (1993)

Huang, W.H., Beevers, K.R.: Topological Map Merging. International Journal of Robotic Research (IJRR) 24(8), 601–613 (2005)

Humphrys, M.: Action Selection methods using Reinforcement Learning. Ph.D. thesis, Trinity Hall, Cambridge (1997)

Humza, R., Scholz, O., Mokhtar, M., Timmis, J., Tyrrell, A.: Towards Energy Homeostasis in an Autonomous Self-Reconfigurable Modular Robotic Organism. In: Proc. of the First International Conference on Adaptive and Self-adaptive Systems and Applications, ADAPTIVE 2009 (2009)

Hyvarinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. Wiley, New York (2001)

Ieropoulos, I., Melhuish, C., Greenman, J., Horsfield, I.: EcoBot-II: An artificial agent with a natural metabolism. International Journal of Advanced Robotic Systems 2(4), 295–300 (2005a)

Ieropoulos, I.A., Greenman, J., Melhuish, C., Hart, J.: Comparative study of three types of microbial fuel cell. Enzyme and Microbial Technology 37(2), 238–245 (2005b)

Ijspeert, A.J., Martinoli, A., Billard, A., Gambardella, L.M.: Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment. Autonomous Robots 11(2), 149–171 (2001)

Inden, B.: Stepwise Transition from Direct Encoding to Artificial Ontogeny in Neuroevolution. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 1182–1191. Springer, Heidelberg (2007)

Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.): RoboCup 2008: Robot Soccer World Cup XII. LNCS, vol. 5399, pp. 614–625. Springer, Heidelberg (2009)

iRobot Corporation. iRobot Roomba 500 Series User's guide. 63 South Avenue, Burlington, MA 01803, USA (2007)

Itti, L., Koch, C.: A comparison of feature combination strategies for saliency-based visual attention systems. SPIE human vision and electronic imaging IV 3644, 473–482 (1999)

Itti, L., Koch, C.: Computational modelling of visual attention. Nature Reviews Neuroscience 2(3), 194–203 (2001)

Itti, L., Koch, C.: Feature combination strategies for saliency-based visual attention systems. Journal of Electronic Imaging 10, 161 (2001)

Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on pattern analysis and machine intelligence 20(11), 1254–1259 (1998)

Jaeger, J., Surkova, S., Blagov, M., Janssens, H., Kosman, D., Kozlov, K., Manu, M.E., Vanario-Alonso, C., Samsonova, M., Sharp, D., Reinitz, J.: Dynamic control of positional information in the early Drosophila embryo. Nature 430, 368–371 (2004)

Jemai, J., Kuerner, T.: Broadband WLAN channel sounder for IEEE 802.11b. IEEE Transactions on Vehicular Technology 57, 3381–3392 (2008)

Jemai, J., Kürner, T.: Towards a Performance Boundary in Calibrating Indoor Ray Tracing Models. EURASIP Journal on Wireless Communication and Networking 8 (2009)

Jemai, J., Schmidt, I., Kürner, T.: UWB channel: from statistical modeling to calibration-based deterministic modelling-Extended Version. Journal of the European Microwave Association, special issue 4, 31–37 (2008)

Jemai, J., Eggers, P., Pedersen, G.F., Kürner, T.: Calibration of a UWB sub-band channel model using simulated annealing. IEEE Transactions on Antennas and Propagation 57, 3439–3443 (2009)

Jones, J.: The Emergence and Dynamical Evolution of Complex Transport Networks from Simple Low-Level Behaviours. Journal of Unconventional Computation (2009) (accepted)

Jong, K., Marchiori, E., Sebag, M.: Ensemble Learning with Evolutionary Computation: Application to Feature Ranking. In: Yao, X., Burke, E., Lozano, J.-A., Smith, J., Guervós, J.M., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1133–1142. Springer, Heidelberg (2004)

Jorgensen, M.W., Ostergaard, E.H., Lund, H.H.: Modular ATRON: Modules for a self-reconfigurable robot. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004 (2004)

Josang, A.: Conditional Reasoning with Subjective Logic. Journal of Multiple-Valued Logic and Soft Computing 15(1), 5–38 (2009)

Judson, O.P., Normark, B.B.: Ancient asexual scandals. Trends in Ecology & Evolution, 11(2), A41–A46 (1996); ISI Document Delivery No.: TT854 Times Cited: 7 Cited Reference Count: 55 ELSEVIER SCI LTD

Kahan, W.: Lectures on cumputational aspects of geometry (1983) (unpublished)

Kalthoff, K.: Pattern formation in early insect embryogenesis - data calling for modification of a recent model. Journal of Cell Science 29(1), 1–15 (1978)

Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a Modular robotic system. IEEE/ASME Transactions on Mechatronics 10(3), 314–325 (2005)

Kapur, J.N., Kesavan, H.K.: Entropy Optimization Principles with Applications. Academic Press, INC., Boston (1992)

Kataoka, N., Kaneko, K.: Functional dynamics I: Articulation process. Physica D 138, 225–250 (2000)

Kavka, C., Roggero, P., Schoenauer, M.: Evolution of Voronoi based fuzzy recurrent controllers. In: Beyer, H.-G. (ed.) Proc. of the conference on Genetic and evolutionary computation (GECCO 2005), pp. 1385–1392. ACM Press, Washington (2005)

Kazadi, S.T., Kondo, E., Cheng, A.: A Robust Centralized Linear Spatial Search Flock. In: Kamel, M. (ed.) Proc. of the 10th IASTED International Conference on Robotics and Applications (IASTED 2004), pp. 52–59 (2004)

Keijzer, F.: Making Decisions does not Suffice for Minimal Cognition. Adaptive Behavior 11(4), 266–269 (2003)

Kelly, I., Holland, O., Melhuish, C.: SlugBot: A Robotic Predator in the Natural World. In: Proc. of the 5th International symposium on artificial life and robotics (AROB 2000), pp. 470–475 (2000)

Kelmar, L., Khosla, P.K.: Automatic generation of kinematics for a reconfigurable modular manipulator system. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 1988), vol. 2, pp. 663–668 (1988)

Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36, 41–50 (2003)

Kernbach, S.: Structural Self-organization in Multi-Agents and Multi-Robotic Systems. Logos Verlag, Berlin (2008)

Kernbach, S., Ricotti, L., Liedke, J., Corradi, P., Rothermel, M.: Study of Macroscopic Morphological Features of Symbiotic Robotic Organisms. In: Proc. of the International Conference on Intelligent Robots and Systems (IROS 2008), Workshop on Self-reconfigurable robots, pp. 18–25 (2008a)

Kernbach, S., Meister, E., Schlachter, F., Jebens, K., Szymanski, M., Liedke, J., Laneri, D., Winkler, L., Schmickl, T., Thenius, R., Corradi, P., Ricotti, L.: Symbiotic Robot Organisms: REPLICATOR and SYMBRION Projects. In: Proc. of Performance Metrics for Intelligent Systems Workshop (PerMIS 2008), pp. 62–69 (2008b)

Kernbach, S., Meister, E., Scholz, O., Humza, R., Liedke, J., Ricotti, L., Jemai, J., Havlik, J., Liu, W.: Evolutionary robotics: The next-generation-platform for on-line and on-board artificial evolution. In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC-2009). IEEE Press, Trondheim (2009a)

Kernbach, S., Hamann, H., Stradner, J., Thenius, R., Schmickl, T., van Rossum, A.C., Sebag, M., Bredeche, N., Yao, Y., Baele, G., Van de Peer, Y., Timmis, J., Mohktar, M., Tyrrell, A., Eiben, A.E., McKibbin, S.P., Liu, W., Winfield, A.F.T.: On adaptive self-organization in artificial robot organisms. In: Proc. of the First IEEE International Conference on Adaptive and Self-adaptive Systems and Applications (IEEE ADAPTIVE 2009) (2009b)

Kernbach, S., Thenius, R., Kernbach, O., Schmickl, T.: Re-Embodiment of Honeybee Aggregation Behavior in Artificial Micro-Robotic System. Adaptive Behavior 17(3), 237–259 (2009c)

Kessin, R.H.: Dictyostelium: Evolution, Cell Biology, and the Development of Multicellularity. Cambridge University Press, Cambridge (2001)

Khan, S., Spudich, J.L., McCray, J.A., Trentham, D.R.: Chemotactic signal integration in bacteria. Proc. of the National Academy of Sciences, USA, Microbiology (1998)

Kimura, M.: The Neutral Theory of Molecular Evolution. Cambridge University Press, Cambridge (1983)

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)

Kitano, H.: Towards a theory of biological robustness. Molecular Sysytems Biology 3(137), 1–7 (2007)

Kitano, H., Oda, K., Kimura, T., Matsuoka, Y., Csete, M., Doyle, J., Muramatsu, M.: Metabolic syndrome and robustness tradeoffs. Diabetes 53(3), S6–S15 (2004)

Kiziltan, Z., Milano, M.: Group-graphs Associated with Row and Column Symmetries of Matrix Models: Some Observations. In: Proc. of the Second International Workshop on Symmetry in Constraint Satisfaction Problems, SymCon 2002 (2002)

Kobayashi, S., Nomizu, K.: Foundations of Differential Geometry, vol. I and II. John Wiley & Sons, New York (1996)

Kobayashi, Y., Hoshi, Y., Hoshino, G., Kasuya, T., Fueki, M., Kuno, Y.: Museum guide robot with three communication modes. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), pp. 3224–3229. IEEE, Nice (2008)

Kolesnikov, A.: Synergetic control theory. Taganrog state university, Taganrog (1994) (in Russian)

Kolter, J.Z., Abbeel, P., Ng, A.: Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, MIT Press, Cambridge (2008)

Kondrashov, A.S.: Classification of hypotheses on the advantage of amphimixis. Journal of Heredity 84(5), 372–387 (1993)

Kornienko, S., Kornienko, O., Priese, J.: Application of multi-agent planning to the assignment problem. Computers in Industry 54(3), 273–290 (2004a)

Kornienko, S., Kornienko, O., Levi, P.: Multi-agent repairer of damaged process plans in manufacturing environment. In: Proc. of the 8th Conf. on Intelligent Autonomous Systems (IAS 8), Amsterdam, NL, pp. 485–494 (2004b)

Kornienko, S., Kornienko, O., Constantinescu, C., Pradier, M., Levi, P.: Cognitive micro-Agents: individual and collective perception in microrobotic swarm. In: Proc. of the Workshop on Agents in real-time and dynamic environments, IJCAI 2005 (2005)

Kornienko, S., Kornienko, O., Nagarathinam, A., Levi, P.: From real robot swarm to evolutionary multi-robot organism. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2007), pp. 1483–1490. IEEE, Singapure (2007)

Korst, P.J.A.M., Velthuis, H.H.W.: The nature of trophallaxis in honeybees. Insectes Sociaux 29(2), 209–221 (1982)

Kouptsov, K.L.: Production-rule complexity of recursive structures. In: Minai, A., Bar-Yam, Y. (eds.) Unifying Themes in Complex Systems IV, Proc. of the Fourth International Conference on Complex Systems, pp. 149–157. Springer, Heidelberg (2008)

Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)

Kozma, R.: Intentional systems: Review of neurodynamics, modeling, and robotics implementation. Physics of Life Reviews 5, 1–21 (2008)

Kulich, M., Rollo, M., Mazl, R., Chudoba, J., Benda, P., Preucil, L., Pechoucek, M.: Multi-robot exploration using multi-agent approach. In: Schilling, K. (ed.) Proc. of the 13th IASTED International Conference on Robotics and Applications/IASTED International Conference on Telematics (RA 2007), pp. 495–500 (2007)

Kumar, S., Bentley, P.J.: Implicit evolvability: An investigation into the evolvability of an embryogeny. In: Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2000). A late-breaking paper, pp. 8–12. Morgan Kaufmann, Las Vegas (2000)

Kumar, V., Rus, D., Sukhatme, G.S.: Networked Robots. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 943–958 (2008)

Kuo, P.D., Leier, A., Banzhaf, W.: Evolving Dynamics in an Artificial Regulatory Network Model. In: Yao, X., Burke, E., Lozano, J.-A., Smith, J., Guervós, J.M., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 571–580. Springer, Heidelberg (2004)

Kuo, W., Zuo, M.J.: Optimal Reliability Modeling: Principles and Applications. Wiley, Chichester (2002)

Kurokawa, H., Kamimura, A., Yoshida, E., Tomita, K., Kokaji, S., Murata, S.: M-TRAN II: Metamorphosis from a Four-Legged Walker to a Caterpillar. In: Proc. of the 2003 IEEE/RSJ Int. Conference on Intelligent Robots and Systems, pp. 2454–2459. IEEE, Las Vegas (2003)

Lafrenz, R., Schanz, M., Kaeppeler, U., Zweigle, O., Rajaie, H., Schreiber, F., Levi, P.: Evaluating Robustness of Coupled Selection Equations Using Sparse Communication. In: Burgard, et al. (eds.) Intelligent Autonomous Systems 10 (IAS 10), pp. 272–277. IOS Press, Amsterdam (2008)

Lakoff, G.: Women, Fire, and Dangerous Things: What Categories Reveal about the Mind. University of Chicago Press, Chicago (1987)

Landis, G.A., Jenkins, P.P.: Dust on Mars: Materials Adherence Experiment results from Mars Pathfinder. In: Proc. of the 26th Photovoltaic Specialists Conference/Conference Record of the Twenty-Sixth IEEE, pp. 865–869. IEEE, Los Alamitos

Lang, C., Hanheide, M., Lohse, M., Wersing, H., Sagerer, G.: Feedback Interpretation based on Facial Expressions in Human Robot Interaction. In: Proc. of the 18th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN 2009), pp. 189–194. IEEE, Toyama (2009)

Lanzi, P.L.: Adaptive Agents with Reinforcement Learning and Internal Memory. In: From Animals to Animats 6: Proc. of Conf. on Simulation of Adaptive Behaviour, pp. 333–342. MIT Press, Cambridge (2000)

Larochelle, H., Erhan, D., Courville, A.C., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proc. of the 24th Annual International Conference on Machine Learning (ICML 2007), pp. 473–480 (2007)

Lau, H.Y.K., Ko, A.W.Y., Lau, T.L.: The design of a representation and analysis method for modular self-reconfigurable robots. Robot. Comput.-Integr. Manuf. 24(2), 258–269 (2008)

LeCun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-Road Obstacle Avoidance through End-to-End Learning. In: Proc. of the Nineteenth Annual Conference on Neural Information Processing Systems, NIPS 2005 (2005)

Ledeczi, A., Karsai, G., Bapty, T.: Synthesis of Self-Adaptive Software. In: Proc. of the IEEE Aerospace Conference 2000, Big Sky, MT (2000)

Lerman, K., Galstyan, A.: A General Methodology for Mathematical Analysis of Multi-Agent Systems. Tech. rept. USC Information Sciences Institute (2001)

Lerman, K., Galstyan, A.: Mathematical model of foraging in a group of robots: Effect of interference. Autonomous Robots 13(2), 127–141 (2002a)

Lerman, K., Galstyan, A.: Two Paradigms for the Design of Artificial Collectives. In: Proc. of the Workshop on Collectives and Design of Complex Systems, NASA/Ames. Springer, Heidelberg (2002b)

Lerman, K., Galstyan, A.: Macroscopic Analysis of Adaptive Task Allocation in Robots. In: Proc. of the International Conference on Intelligent Robots and Systems, IROS 2003 (2003)

Lerman, K., Galstyan, A., Hogg, T.: Mathematical Analysis of Multi-Agent Systems. e-print arXiv:cs/0404002v1, arXiv.org (2004)

Lerman, K., Jones, C., Galstyan, A., Matarić, M.: Analysis of Dynamic Task Allocation in Multi-Robot Systems. International Journal of Robotics Research 25(3), 225–242 (2006)

Levi, P.: Architectures of individual and distributed autonomous agents. In: Proc. of the 2nd Int. Conf. on Intelligent Autonomous Systems, IAS-2 (1989)

Levi, P.: Development of Evolutionary and Self-Assembling Robot-Organisms. In: Proc. of the 20th Anniversary IEEE MHS- 2009 and Micro-Nano Global COE, Nagoya, Japan, pp. 1–6 (2009)

Levi, P., Schanz, M., Kornienko, S., Kornienko, O.: Application of Order Parameter Equation for the Analysis and the Control of Nonlinear Time Discrete Dynamical Systems. Int. Journal of Bifurcation and Chaos 9(8), 1619–1634 (1999)

Li, G., Lin, K.-C., Xia, Z.: Rule-Based Control of Collaborative Robots. In: Proc. of the 16th International Conference on Artificial Reality and Telexistence, pp. 68–72. IEEE Computer Society, Los Alamitos (2006)

Linden, D., Reddy, T.B.: Handbook of Batteries, 3rd edn. McGraw-Hill, New York (2002)

Lipson, H.: Uncontrolled Engineering: A Review of S. Nolfi and D. Floreano's Evolutionary Robotics. Artificial Life 4(7), 419–424 (2000)

Littman, M., Sutton, R., Singh, S.: Predictive Representation of States. In: Proc. of the Sixteenth Annual Conference on Neural Information Processing Systems (NIPS 2002), vol. 14, pp. 1555–1561 (2002)

Liu, W.: Design and Modelling of Adaptive Foraging in Swarm Robotic Systems. Ph.D. thesis, University of the West of England, Bristol (September 2008)

Liu, W., Winfield, A.F.T., Sa, J.: Modelling Swarm Robotic Systems: A Case Study in Collective Foraging. In: Proc. of the Conference Towards Autonomous Robotics Systems (TAROS 2007), pp. 25–32 (2007)

Liu, W., Winfield, A.F.T., Sa, J.: A macroscopic probabilistic model of adaptive foraging in swarm robotics systems. In: Proc. of the 6th Vienna International Conference on Mathematical Modelling (MATHMOD 2009), Special Session on Modelling the Swarm (2009)

Llorà, X., Sastry, K., Goldberg, D.E., Gupta, A., Lakshmi, L.: Combating user fatigue in iGAs: partial ordering, support vector machines, and synthetic fitness. In: Beyer, H.-G., O'Reilly, U.-M. (eds.) Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2005), pp. 1363–1370. ACM, New York (2005)

Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.M.: A survey for the quadratic assignment problem. European Journal of Operational Research 176(2), 657–690 (2007)

Lorch, J.R., Smith, A.J.: Software Strategies for Portable Computer Energy Management. IEEE Personal Communications 5, 60–73 (1998)

Lorenz, K.: Vergleichende Bewegungsstudien an Anatiden. J. Ornithol. 89, 194–293 (1941)

Lorincz, K., Welsh, M.: MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In: Strang, T., Linnhoff-Popien, C. (eds.) LoCA 2005. LNCS, vol. 3479, pp. 63–82. Springer, Heidelberg (2005)

Luh, J.Y.S., Walker, M.W., Paul, R.P.C.: On-Line Computational Scheme for Mechanical Manipulators. ASME Journal of Dynamic Systems, Measurement and Control 102(2), 69–76 (1980)

Luminary Micro. LM3S8962 Microcontroller Data Sheet. Rev ds-lm3s8962-3447 edn. (2008)

Luminary Micro. LM3S8970 Microcontroller Data Sheet. Rev. ds-lm3s8970-6462 edn. (2009)

Lund, H.H., Hallam, J., Lee, W.-P.: Evolving Robot Morphology (1997)

Lungarella, M., Metta, G., Pfeifer, R., Sandini, G.: Developmental robotics: a survey. Connect. Sci. 15(4), 151–190 (2003)

Lyder, A., Garcia, R., Stoy, K.: Mechanical design of odin, an extendable heterogeneous deformable modular robot. In: International Conference on Intelligent Robots and Systems (IROS 2008), pp. 883–888 (2008)

Maes, P.: How to do the right thing. Connect. Sci. J. Spec. Issue Hybrid Systems 1, 291–323 (1990)

Maley, C.C.: Four Steps Toward Open-Ended Evolution. In: Proc. of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 1336–1343. Morgan Kaufmann, San Francisco (1999)

Maniadakis, M., Trahanias, P.: Design and Integration of Partial Brain Models Using Hierarchical Cooperative CoEvolution. In: Proc. of the International Conference on Cognitive Modelling (ICCM 2006), pp. 196–201 (2006)

Manseur, R., Doty, K.L.: A complete kinematic analysis of four-revolute-axis robot manipulators. ASME Mechanisms and Machines 27(5), 575–586 (1992a)

Manseur, R., Doty, K.L.: Fast inverse kinematic analysis of five-tevolute-axis robot manipulators. ASME Mechanisms and Machines 27(5), 587–597 (1992b)

Martinoli, A.: Optimization of Swarm Robotic Systems via Macroscopic Models. In: Proc. of the Second Int. Workshop on Multi-Robots Systems, pp. 181–192 (2003)

Martinoli, A., Easton, K.: Modeling Swarm Robotic Systems. Springer Tracts in Advanced Robotics 5, 297–306 (2003)

Martinoli, A., Ijspeert, A.J., Gambardella, L.M.: Understanding Collective Aggregation Mechanisms: From Probabilistic Modelling to Experiments with Real Robots. Robotics and Autonomous Systems 29(1), 51–63 (1999)

Martinoli, A., Easton, K., Agassounon, W.: Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation. International Journal of Robotics Research 23(4), 415–436 (2004)

Maslow, A.H.: Toward a Psychology of Being. Wiley, Hoboken (1998)

Mataric, M.J., Cliff, D.: Challenges in evolving controllers for physical robots. Robotics and Autonomous Systems 19(1), 67–83 (1996)

Mataric, M.J.: Situated Robotics. Encyclopedia of Cognitive Science. Nature Publishers Group, Macmillian Reference Ltd. (2002)

Matsumoto, Y., Inaba, M., Inoue, H.: Visual navigation using view-sequenced route representation. In: Proc. of the IEEE International Conference Robotics and Automation, vol. 1, pp. 83–88 (1996)

Mattiussi, C., Floreano, D.: Analog Genetic Encoding for the Evolution of Circuits and Networks. IEEE Transactions on Evolutionary Computation 11(5), 596–607 (2007)

Matzinger, P.: An Innate Sense of Danger. Seminars in Immunology 10(5), 399–415 (1997)

Mazl, R., Preucil, L.: Building a 2D environment map from laser range-finder data. In: Proc. of the IEEE Intelligent Vehicles Symposium, pp. 290–295 (2000)

McCormack, J.: Interactive Evolution of L-System Grammars for Computer Graphics Modelling. In: Green, D.G., Bossomaier, T. (eds.) Complex systems: from biology to computation, pp. 118–130. IOS Press, Amsterdam (1993)

McFarland, D., Bösser, T.: Intelligent behavior in animals and robots. Complex adaptive systems. A Bradford book, MIT Press, Cambridge and London (1993)

McFarland, D., Spier, E.: Basic Cycles, Utility and Opportunism in Self-Sufficient Robots. In: Robotics and Autonomous Systems, pp. 179–190 (1997)

McLean, K.C., Pasupathi, M., Pals, J.L.: Selves Creating Stories Creating Selves: A Process Model of Self-Development. Personality and Social Psychology Review 11(3), 262–278 (2007)

Mei, Y., Lu, Y.-H., Hu, Y.C., Lee, C.S.G.: A case study of mobile robot's energy consumption and conservation techniques. In: Proc. of the 12th International Conference Advanced Robotics (ICAR 2005), pp. 492–497 (2005)

Meinhardt, H.: The Algorithmic Beauty of Sea Shells. Springer, Heidelberg (1995)

Meinhardt, H., Gierer, A.: Pattern Formation by Local Self-activation and Lateral Inhibition. Bioessays 22, 753–760 (2000)

Melhuish, C., Kubo, M.: Collective Energy Distribution: Maintaining the Energy Balance in Distributed Autonomous Robots using Trophallaxis. In: Distributed Autonomous Robotic Systems 6, pp. 275–284. Springer, Heidelberg (2007)

Michaud, S., Schneider, A., Bertrand, R., Lamon, P., Siegwart, R., Van Winnendael, M., Schiele, A.: SOLERO: Solar-Powered Exploration Rover. In: Proc. of the 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2002), p. 8 (2002)

Michiels, N.K., Newman, L.J.: Sex and violence in hermaphrodites. Nature. 391(6668), 647–647 (1998); ISI Document Delivery No.: YW872 Times Cited: 43 Cited Reference Count: 7 Macmillan Magazines Ltd.

Microchip. PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers. Ds30292c edn. (2001)

Mikhailov, A.S., Calenbuhr, V.: From Cells to Societies. Springer, Heidelberg (2002)

Miller, J.F., Banzhaf, W.: Evolving the Program for a Cell: From French Flags to Boolean Circuits. In: On Growth, Form and Computers, pp. 278–302. Academic Press, London (2003)

Minsky, M.: Frame-system theory. In: Johnson-Laird, P.N., Wason, P.C. (eds.) Thinking. Readings in cognitive science. Cambridge University Press, Cambridge (1977)

Mokhtar, M., Bi, R., Timmis, J., Tyrrell, A.M.: A modified dendritic cell algorithm for on-line error detection in robotic systems. In: Proc. of the IEEE Congress on Evolutionary Computation, CEC 2008 (2008)

Molenaar, J., de Weger, J.G., van de Water, W.: Mappings of Grazing Impact Oscillators. Nonlinearity, 301–321 (2001)

Mondada, F., Guignard, A., Colot, A., Floreano, D., Deneubourg, J.-L., Gambardella, L., Nolfi, S., Dorigo, M.: SWARM-BOT: A New Concept of Robust All-Terrain Mobile Robotic System. Tech. rept. Swiss Federal Institute of Technology, Lausanne, Switzerland (2002)

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S.E., Zufferey, J.-C., Floreano, D., Martinoli, A.: The e-puck, a Robot Designed for Education in Engineering. In: Proc. of the 9th Conference on Autonomous Robot Systems and Competitions, vol. 1, pp. 59–65 (2009)

Montemerlo, M., Thrun, S., Dahlkamp, H., Stavens, D., Strohband, S.: Winning the DARPA Grand Challenge with an AI robot. In: Proc. of the Twenty-First National Conference on Artificial Intelligence, AAAI 2006 (2006)

Mueller, C.T., Levin, M.: The use of classical conditioning in planaria to investigate a non-neuronal memory mechanism (2002)

Müller, G.B.: Evo-Devo: extending the evolutionary synthesis. Nature Reviews Genetics 8, 943–949 (2007)

Müller, S.C., Mair, T., Steinbock, O.: Travelling waves in yeast extract and in cultures of *Dictyostelium discoideum*. Biophysical Chemistry 72, 37–47 (1998)

Munos, R., Moore, A.W.: Variable Resolution Discretization in Optimal Control. Machine Learning 49(2-3), 291–323 (2002)

Murata, S., Kurokawa, H.: Self-Reconfigurable Robots: Shape-Changing Cellular Robots Can Exceed Conventional Robot Flexibility. IEEE Robotics & Automation Magazine (2007)

Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)

Nään, A.M., Lemon, B., Tjian, R.: Transcriptional coactivator complexes. Annual Review of Biochemistry 70, 475–501 (2001)

Nagel, T.: What Is It Like To Be a Bat? Philosophical Review 83(4), 435–450 (1974)

Narendra, K.S., Annaswamy, A.M.: Stable Adaptive Systems. Prentice-Hall, Englewood Cliffs (1989)

Neal, M.J., Timmis, J.: Timidity: A useful mechanism for robot control? Informatica 4(27), 197–204 (2003)

Nehaniv, C., Dautenhahn, K. (eds.): Imitation and Social Learning in Robots, Humans and Animals. Cambridge University Press, Cambridge (2007)

Nehmzow, U.: Physically Embedded Genetic Algorithm Learning in Multi-Robot Scenarios: The PEGA Algorithm. In: Prince, C.G., Demiris, Y., Marom, Y., Kozima, H., Balkenius, C. (eds.) Proc. of The Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. Lund University Cognitive Studies, vol. 94. LUCS, Edinburgh (2002)

Nembrini, J.: Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots. PhD Thesis, University of the West of England, Bristol, UK (2005)

Nembrini, J., Winfield, A.F.T., Melhuish, C.: Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots. In: From Animals to Animats (SAB 2002), pp. 373–382. MIT Press, Cambridge (2002)

Nepomnyashchikh, V.A., Popov, E.E., Red'ko, V.G.: A bionic model of adaptive searching behavior. Journal of Computer and Systems Sciences International 47(1), 78–85 (2008)

Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: Langley, P. (ed.) Proc. of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 663–670. Morgan Kaufmann, San Francisco (2000)

Ngo, T.D., Schiøler, H.: Sociable Robots through Self-maintained Energy. International Journal of Advanced Robotic Systems 3(4), 313–322 (2006)

Nilsson, M.: Heavy-duty connectors for self-reconfiguring robots. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2002), vol. 4, pp. 4071–4076. IEEE, Los Alamitos (2002)

Nisbet, R.M., Muller, E.B., Lika, K., Kooijman, S.A.L.M.: From molecules to ecosystems through dynamic energy budget models. Journal of Animal Ecology 69(6), 913–926 (2000)

Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. The MIT Press, Cambridge (2000a)

Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press, Cambridge (2000b)

Nolfi, S., Parisi, D.: Auto-teaching: networks that develop their own teaching input. In: Deneubourg, J., Bersini, H., Goss, S., Nicolis, G., Dagonnier, R. (eds.) Proc. of the Second European Conference on Artificial Life (1993)

Nolfi, S., Parisi, D.: Learning to adapt to changing environments in evolving neural networks. Adaptive Behavior 5(1), 75–98 (1997)

Nuechtera, A., Hertzberga, J.: High-speed laser localization for mobile robots. Elsevier Robotics and Autonomous Systems 51, 275–296 (2005)

Nüsslein-Volhard, C.: Das Werden des Lebens. Verlag C.H. Beck, Müchen (2004)

NXP. LPC2141/42/44/46/48 Single-chip 16-bit/32-bit microcontrollers. Rev. 04 edn. (November 2008)

Ogata, T., Sugano, S.: Emotional communication robot: WAMOEBA-2R emotion model and evaluation experiments. In: Proc. of the International Conference on Humanoid Robots, Humanoid 2000 (2000)

Orin, D.E., McGhee, R.B., Vukobratovic, M., Hartoch, G.: Kinematic and Kinetic Analyses of Open-chain Linkages Utilizing Newton-Euler Methods. Mathematical Bioschiences 43, 107–130 (1979)

Osbourne, P.V., Whitaker, H.P., Kezer, A.: New Developments in the Design of Model Reference Adaptive Control Systems. Inst. Aeronautical Services (1961)

Oudeyer, P.-Y., Kaplan, F.: Intelligent Adaptive Curiosity: a source of Self-Development. Lund University Cognitive Studies, pp. 127–130 (2004)

Owens, N., Timmis, J., Greensted, A., Tyrrell, A.: On Immune Inspired Homeostasis for Electronic Systems. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 216–227. Springer, Heidelberg (2007)

Owens, N., Greensted, A., Timmis, J., Tyrrell, A.: T Cell Receptor Signalling inspired Kernel Density Estimation and Anomaly Detection. In: Andrews, P., Timmis, J., Owens, N.D.L., Aickelin, U., Hart, E., Hone, A., Tyrrell, A. (eds.) ICARIS 2009. LNCS, vol. 5666, pp. 122–155. Springer, Heidelberg (2009)

Owens, N., Timmis, J., Greensted, A., Tyrrell, A.: Elucidation of T Cell Signalling Models. Journal of Theoretical Biology (2010) (accepted)

Paden, B.: Kinematics and Control Robot Manipulators. Ph.D. thesis, Department of Electrical Engineering and Computer Sciences, University of California, Barkeley (1986)

Parker, G.A., Baker, R.R., Smith, V.G.F.: The origin and evolution of gamete dimorphism and the male-female phenomenon. Journal of Theoretical Biology 36(3), 529–553 (1972)

Parker, L.E.: Multiple Mobile Robot Systems. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 921–941 (2008)

Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. Autonomous Robots 11(3), 319–324 (2001)

Penrose, R.: Shadows of the Mind: A Search for the Missing Science of Consciousness. Oxford University Press, Oxford (1994)

Penrose, R.: The Road to Reality. Alfred A. Knopf, Inc., New York (2006)

Pezzulo, G.: A Study of Off-Line Uses of Anticipation. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS (LNAI), vol. 5040, pp. 372–382. Springer, Heidelberg (2008)

Pfeifer, R.: Building "Fungus Eaters": Design Principles of Autonomous Agents. In: Mataric, M.J., Wilson, S.W., Meyer, J.-A. (eds.) Proc. of the Fourth International Conference on Simulation of Adaptive Behavior (SAB 1996), pp. 3–12. MIT Press, Cambridge (1996)

Pfeifer, R., Bongard, J.C.: How the Body Shapes the Way We Think: A New View of Intelligence (Bradford Books). The MIT Press, Cambridge (2006)

Pfeifer, R., Iida, F., Bongard, J.: New robotics: Design principles for intelligent systems. Artificial Life 11(1-2), 99–120 (2005a)

Pfeifer, R., Iida, F., Bongard, J.: New Robotics: Design Principles for Intelligent Systems. Artificial Life 11(1-2), 99–120 (2005b)

Pfeifer, R., Iida, F., Gomez, G.: Morphological computation for adaptive behavior and cognition. International Congress Series (2006)

Piaget, J.: La construction du rel chez l'enfant. Delachaux et Niestl (1937)

Pierce, J.R.: An Introduction to Information Theory. Symbols, Signals and Noise. Dover Publications, Inc., Cambridge (1980)

Pomerleau, D.A.: ALVIN-N: An Autonomous Land Vehicle In a Neural Network. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems. Morgan Kaufmann, San Francisco (1989)

Popescu-Belis, A.: An adaptive multi-agent system based on "neural Darwinism". In: Proc. of the first international conference on Autonomous agents (AGENTS 1997), pp. 484–485. ACM, New York (1997)

Pradier, M.: Collective Classification in a Swarm of Microrobots. Master Thesis, University of Stuttgart, Germany (2005)

Precup, D., Sutton, R.S., Paduraru, C., Koop, A., Singh, S.P.: Off-policy Learning with Options and Recognizers. Advances in Neural Information Processing Systems 18, 1097–1104 (2006)

Prescott, T.J., Bryson, J.J., Seth, A.K. (eds.): Theme Issue on Models of Natural Action Selection. Philosophical Transactions of the Royal Society. B. 362(1485) (2007)

Prescott, T.T.: Action Selection. Scholarpedia 3(2), 2705 (2008)

Prigogine, I.: Non-Equilibrium Statistical Mechanics. Wiley Interscience, New York (1962)

Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM 2000), pp. 32–43 (2000)

Prokhorov, D.V., Wunsch, D.C.: Adaptive critic designs. IEEE Transactions on Neural Networks 8(5), 997–1007 (1997)

Prusinkiewicz, P., Hanan, J.: Lindenmayer Systems, Fractals, and Plants. Springer, Heidelberg (1980)

Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience, Hoboken (1994)

Qiao, G., Lu, R., McLean, C.: Flexible manufacturing systems for mass customisation manufacturing. International Journal of Mass Customisation 1(2/3), 374–393 (2006)

Quayle, A.P., Bullock, S.: Modelling the evolution of genetic regulatory networks. Journal of Theoretical Biology 238, 737–753 (2006)

Quick, T., Nehaniv, C.L., Dautenhahn, K., Roberts, G.: Evolving embodied genetic regulatory network-driven control systems. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 266–277. Springer, Heidelberg (2003)

Raibert, M., Blankespoor, K., Nelson, G., Playter, R.: BigDog, the Rough-Terrain Quaduped Robot. In: IFAC workshop (2008)

Rasmussen, S., Chen, L., Deamer, D., Krakauer, D.C., Packard, N.H., Stadler, P.F., Bedau, M.A.: EVOLUTION: Transitions from Nonliving to Living Matter. Science 303(5660), 963–965 (2004)

Ratliff, N.D., Silver, D., Bagnell, J.A.: Learning to search: Functional gradient techniques for imitation learning. Auton. Robots 27(1), 25–53 (2009)

Ray, S.: Survival of olfactory memory through metamorphosis in the fly Musca domestica. Neuroscience Letters 259(1), 37–40 (1999)

Red'ko, V.G.: Estimation of evolution rate in Eigen's and Kuhn's models (in Russian). Biofizika 31(3), 511–516 (1986)

Red'ko, V.G.: Spin glasses and evolution (in Russian). Biofizika 35(5), 831–834 (1990)

Red'ko, V.G.: Towards modeling cognitive evolution. In: Proc. of the Fifth International Conference on Neural Networks and Artificial Intelligence, pp. 17–21 (2008)

Red'ko, V.G., Beskhlebnova, G.A.: Model of adaptive behavior of autonomous agents in two-dimensional cellular environment (in Russian). In: Proc. of Russian Conference on Neuroinformatics, pp. 169–177 (2009)

Red'ko, V.G., Tsoy, Y.R.: Estimation of the efficiency of evolution algorithms. Doklady Mathematics (Doklady Akademii Nauk) 72(2), 810–813 (2005)

Red'ko, V.G., Mosalov, O.P., Prokhorov, D.V.: A model of Baldwin effect in populations of self-learning agents. In: Proc. of the International Joint Conference on Neural Networks (IJCNN 2005), pp. 1355–1360 (2005a)

Red'ko, V.G., Mosalov, O.P., Prokhorov, D.V.: A model of evolution and learning. Neural Networks 18(5-6), 738–745 (2005b)

Reil, T.: Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In: Floreano, D., Mondada, F. (eds.) ECAL 1999. LNCS, vol. 1674. Springer, Heidelberg (1999)

Renesas, SH-2 SH7047 Group Hardware Manual. Rev.2.00 edn. (2004)

Renesas, H8/3687 Group Hardware Manual. Rev.5.00 edn. (2005)

REPLICATOR. 2008-2012. REPLICATOR: Robotic Evolutionary Self-Programming and Self-Assembling Organisms, 7th Framework Programme Project No FP7-ICT-2007.2.1. European Communities

Richerson, P.J., Boyd, R.: Built for Speed, Not for Comfort. History and Philosophy of the Life Sciences 23, 423–462 (2001)

Roggen, D., Federici, D.: Multi-cellular development: is there scalability and robustness to gain? In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 391–400. Springer, Heidelberg (2004)

Rohrs, C.E., Valavani, L., Athans, M., Stein, G.: Robustness of Continuous-time Adaptive Control Algorithms in the Presence of Unmodeled Dynamics. IEEE Transactions on Automatic Control 30(9), 881–889 (1985)

Rosset, S.: Model selection via the AUC. In: Proc. of the Twenty-first International Conference (ICML 2004). ACM International Conference Proceeding Series, vol. 69, ACM, New York (2004)

Rosslenbroich, B.: The theory of increasing autonomy in evolution: a proposal for understanding macroevolutionary innovations. Biology & Philosophy (2009)

Rouff, C., Truszkowski, W., Rash, J., Hinchey, M.: Formal Approaches to Intelligent Swarms. In: IEEE/NASA Software Engineering Workshop (SEW 2003), pp. 51–57. IEEE Computer Society, Los Alamitos (2003)

Roweis, S.T., Saul, L.K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290(5500), 2323–2326 (2000)

Rozenberg, G., Salomaa, A.: The mathematical theory of L systems. Academic Press, New York (1980)

Ruiz-Mirazo, K., Umerez, J., Moreno, A.: Enabling conditions for open-ended evolution. Biology & Philosophy 23, 67–85 (January 2008)

Rumelhart, D.E., Hinton, G.E., Williams, R.G.: Learning representation by back-propagating error. Nature 323(6088), 533–536 (1986)

Şahin, E., Winfield, A.: Special issues on swarm robotics. Swarm Intelligence 2(2-4), 69–72 (2008)

Salemi, B., Shen, W.-M.: Distributed Behavior Collaboration for Self-Reconfigurable Robots. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2004), pp. 4178–4183 (2004)

Salemi, B., Moll, M., Shen, W.M.: SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), pp. 3636–3641. IEEE, Los Alamitos (2006)

Salomon, R.: Increasing adaptivity through evolution strategies. In: From Animals to Animats 4: Proc. of the Fourth International Conference on Simulation of Adaptive Behavior, pp. 411–420. MIT Press, Cambridge (1996)

SAP. Adaptive Manufacturing: enabling the lean six sigma enterprises. SAP (2005)

Sastry, S., Bodson, M.: Adaptive Control. Prentice-Hall, Englewood Cliffs (1989)

Saxena, A., Wong, L.L.S., Ng, A.Y.: Learning Grasp Strategies with Partial Shape Information. In: Proc. of the Twentieth Innovative Applications of Artificial Intelligence Conference (AAAI 2008), pp. 1491–1494 (2008a)

Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic Grasping of Novel Objects using Vision. International Journal of Robotics Research (2008b)

Scassellati, B.: Building Behaviors Developmentally: A New Formalism. In: Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998), AAAI Press, Menlo Park (1998)

Schmickl, T., Crailsheim, K.: A Navigation Algorithm for Swarm Robotics Inspired by Slime Mold Aggregation. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433, pp. 1–13. Springer, Heidelberg (2007)

Schmickl, T., Crailsheim, K.: Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm. Autonomous Robots 25(1-2), 171–188 (2008)

Schmickl, T., Crailsheim, K.: Modelling a Hormone-Based Robot Controller. In: Troch, I., Breitenecker, F. (eds.) Proc. of the 6th Vienna International Conference on Mathematical Modelling, MATHMOD 2009 (2009)

Schmickl, T., Möslinger, C., Thenius, R., Crailsheim, K.: Bio-inspired navigation of autonomous robots in heterogenous environments. International Journal of Factory Automation, Robotics and Soft Computing 3, 164–170 (2007)

Schmickl, T., Möslinger, C., Crailsheim, K.: Collective perception in a robot swarm. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433, pp. 144–157. Springer, Heidelberg (2007)

Schmickl, T., Möslinger, C., Thenius, R., Crailsheim, K.: Individual adaptation allows collective path-finding in a robotic swarm. International Journal of Factory Automation, Robotics and Soft Computing, 102–108 (2007c)

Schmickl, T., Stradner, J., Hamann, H., Crailsheim, K.: Major Feedbacks that Support Artificial Evolution in Multi-Modular Robotics. In: Proc. of the IROS 2009 workshop Exploring New Horizons in Evolutionary Design of Robots (EvoDeRob). LNCS. Springer, Heidelberg (2009)

Schmitt, A., Bender, J.: Impulse-Based Dynamic Simulation of Multibody Systems: Numerical Comparison with Standard Methods. In: Proc. of Automation of Discrete Production Engineering, pp. 324–329 (2005)

Schut, M., Haasdijk, E., Eiben, A.E.: What is Situated Evolution? In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC 2009). IEEE Press, Trondheim (2009)

Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, New York (1995)

Secker, A., Freitas, A., Timmis, J.: AISEC An Artificial Immune System for Email Classification. In: Proc. of the Congress on Evolutionary Computation, pp. 131–139 (2003)

Selig, J.M.: Geometric Fundamentals of Robotics. Monographs in Computer Science. Springer, Heidelberg (2004)

Shafer, G.: A Mathematical Theory of Evidence. Princeton Univ. Press, Princeton (1976)

Shanahan, M.: A cognitive architecture that combines internal simulation with a global workspace. Consciousness and Cognition 15(2), 433–449 (2006)

Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 79–423 (1948)

Shapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. In: Fisher, D.H. (ed.) Proc. of the Fourteenth International Conference on Machine Learning (ICML 1997), pp. 322–330. Morgan Kaufmann, San Francisco (1997)

Shen, W.-M., Salemi, B., Will, P.: Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots. IEEE Transactions on Robotics and Automation, 700–712 (2002)

Shen, W.-M., Will, P., Galstyan, A., Chuong, C.-M.: Hormone-Inspired Self-Organization and Distributed Control of Robotic Swarms. Autonomous Robots 17(1), 93–105 (2004)

Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics. Springer, Heidelberg (2008)

Siegert, F., Weijer, C.J.: Three-dimensional scroll waves organize Dictyostelium slugs. Proc. of the National Academy of Science, USA, Developmental Biology 89, 6433–6437 (1992)

Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, Boca Raton (1986)

Simmons, R.: Coordinating planning, perception, and action for mobile robots. SIGART Bull. 2(4), 156–159 (1991)

Smith, J.M.: The theory of games and the evolution of animal conflicts. J. Theor. Biol. 47(1), 209–222 (1974)

Sneller, M.: Granuloma Formation, Implications for the Pathogenesis of Vasculitis. Cleveland Clinic Journal of Medicine 69(2), 40–43 (2002)

Sofge, D.A., Potter, M.A., Bugajska, M.D., Schultz, A.C.: Challenges and Opportunities of Evolutionary Robotics. In: Proc. of the Second International Conference on Computational Intelligence, Robotics and Autonomous Systems, CIRAS 2003 (2003)

Spector, L., Klein, J., Feinstein, M.: Division blocks and the open-ended evolution of development, form, and behavior. In: Proc. of the 9th annual conference on Genetic and evolutionary computation (GECCO 2007), pp. 316–323. ACM, New York (2007)

Spier, E., Mcfarland, D.: Possibly Optimal Decision-Making Under Self-sufficiency and Autonomy. Journal of Theoretical Biology 189, 189–317 (1986)

Standish, R.K.: Open-Ended Artificial Evolution. Int. journal of computational intelligence and application 3, 167 (2003)

Steels, L.: When are robots intelligent autonomous agents? Robotics And Autonomous Systems 15(1), 3–9 (1995)

Steinbock, O., Siegert, F., Müller, S.C., Weijer, C.J.: Three-dimensional waves of excitation during *Dictyostelium* morphogenesis. Proc. of the National Academy of Science, USA, Biophysics 90, 7332–7445 (1993)

Štěpán, P., Kulich, M., Přeučil, L.: Robust Data Fusion with Occupancy Grid. IEEE Transactions on Systems, Man, and Cybernetics: Part C 35(1), 106–115 (2005)

Stepanenko, Y., Vukobratovic, M.: Dynamics of Articulated Open-chain Active Mechanisms. Mathematical Biosciences 28, 137–170 (1976)

Stoy, K.: How to Construct Dense Objects with Self-Reconfigurable Robots. In: Christensen, H. (ed.) Proc. of the European Robotics Symposium 2006. Springer tracts in advanced robotics, vol. 22, pp. 27–37. Springer, Heidelberg (2006)

Støy, K., Shen, W.-M., Will, P.: Using role-based control to produce locomotion in chain-type self-reconfigurable robots. IEEE/ASME Trans. on Mechatronics 7(4), 410–417 (2002)

Stradner, J., Hamann, H., Schmickl, T., Thenius, R., Crailsheim, K.: Evolving a novel bio-inspired controller in reconfigurable robots. In: Proc. of the 10th European Conference on Artificial Life (ECAL 2009). LNCS. Springer, Heidelberg (2009)

Sugawara, K., Sano, M.: Cooperative acceleration of task performance: foraging behavior of interacting multi-robots system. Phys. D 100(3-4), 343–354 (1997)

Sugawara, K., Watanabe, T.: A study on foraging behavior of simple multi-robot system. In: Proc. of the 28th Annual Conference of the IEEE Industrial Electronics Society (IECON 2002), vol. 4, pp. 3085–3090 (2002)

Sugawara, K., Sano, M., Yoshihara, I., Abe, K., Watanabe, T.: Foraging behaviour of multi-robot system and emergence of swarm intelligence. In: Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 1999), vol. 3, pp. 257–262 (1999)

Sumpter, D.J.T., Pratt, S.C.: A modelling framework for understanding social insect foraging. Behavioral Ecology and Sociobiology 53, 131–144 (2003)

Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

Svab, J., Krajnik, T., Preucil, L.: FPGA-based Speeded Up Robust Features. In: Proc. of the IEEE International Conference on Technologies for Practical Robot Applications, TePRA 2009 (2009)

Syed, U., Schapire, R.: A Game-Theoretic Approach to Apprenticeship Learning. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 1449–1456. MIT Press, Cambridge (2008)

Syed, U., Bowling, M.H., Schapire, R.E.: Apprenticeship learning using linear programming. In: Proc. of the Twenty-Fifth International Conference on Machine Learning (ICML 2008). ACM International Conference Proceeding Series, vol. 307, pp. 1032–1039. ACM, New York (2008)

Szymanski, M., Wörn, H.: JaMOS - A MDL2e based Operating System for Swarm Micro Robotics. In: IEEE Swarm Intelligence Symposium, pp. 324–331 (2007)

Szymanski, M., Fischer, J., Wörn, H.: Investigating the Effect of Pruning on the Diversity and Fitness of Robot Controllers based on MDL2e during Genetic Programming. In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC-2009), IEEE Press, Trondheim (2009a)

Szymanski, M., Winkler, L., Laneri, D., Schlachter, F., van Rossum, A.C., Schmickl, T., Thenius, R.: SymbricatorRTOS: A Flexible and Dynamic Framework for Bio-Inspired Robot Control Systems and Evolution. In: Tyrrell, A. (ed.) Proc. of the IEEE Congress on Evolutionary Computation (IEEE CEC-2009). IEEE Press, Trondheim (2009b)

Tani, J.: Self-organization of neuronal dynamical structures through sensory-motor experiences of robots. In: Proc. of the Int. Conf. on Humanoid Robots, Workshop on Intelligence Dynamics (2005)

Tani, J., Ito, M., Sugita, Y.: Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB. Neural Networks 17, 1273–1289 (2004)

Taylor, M.E., Whiteson, S., Stone, P.: Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In: Proc. of the 8th annual conference on Genetic and evolutionary computation (GECCO 2006), pp. 1321–1328. ACM, New York (2006)

Tenenbaum, J.B., de Silva, V., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290, 2319 (2000)

Teo, J.: Darwin + Robots = Evolutionary Robotics: Challenges in Automatic Robot Synthesis. In: Proc. of the 2nd International Conference on Artificial Intelligence in Engineering and Technology (ICAIET 2004), vol. 1, pp. 7–13 (2004)

Thenius, R., Schmickl, T., Crailsheim, K.: Novel Concept of Modelling Embryology for Structuring an Artificial Neural Network. In: Troch, I., Breitenecker, F. (eds.) Proc. of the 6th Vienna International Conference on Mathematical Modelling, MATHMOD 2009 (2009)

Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press, Cambridge (2005a)

Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, Cambridge (2005b)

Tianyun, H., Xiaonan, W., Xuebo, C., Wangbao, X.: Architecture analysis and design of swarm robot systems based on the multi-tasks. In: Proc. of the 27th Chinese Control Conference (CCC 2008), pp. 300–304 (2008)

Timmis, J., de Lemos, R., Ayara, M., Duncan, R.: Towards Immune Inspired Fault Tolerance in Embedded Systems. In: Wang, L., Rajapakse, J., Fukushima, K., Lee, S., Yao, X. (eds.) Proc. of the 9th International Conference on Neural Information Processing, pp. 1459–1463. IEEE, Los Alamitos (2002)

Timmis, J., Andrews, P., Owens, N., Clark, E.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1(1), 5–26 (2008)

Timmis, J., Nealz, M., Thorniley, J.: An Adaptive Neuro-Endocrine System for Robotic Systems. In: Proc. of the IEEE Workshop on Robotic Intelligence in Informationally Structured Space, RiiSS 2009 (2009)

Tolman, E.C.: Cognitive maps in rats and men. Psychological Review 55, 189–208 (1948)

Tourassis, V.D., Ang, M.H.: Task decoupling in robot manipulators Journal. Journal of Intelligent and Robotic Systems 14(3), 283–302 (1995)

Toussaint, M.: A sensorimotor map: Modulating lateral interactions for anticipation and planning. Neural computation 18(5), 1132–1155 (2006)

Trianni, V.: Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots. Springer, Heidelberg (2008)

Tribelhorn, B., Dodds, Z.: Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2007), pp. 1393–1399 (2007)

Truszkowski, W., Hinchey, M., Rash, J., Rouff, C.: NASA's Swarm Missions: The Challenge of Building Autonomous Software. IT Professional 6(5), 47–52 (2004)

Tsoy, Y.R., Red'ko, V.G.: Efficiency of evolutionary search in quasispecies model. Fuzzy Systems and Soft Computing 1(1) (2006)

Tsoy, Y.R., Red'ko, V.G.: Estimation of the evolution speed for the quasispecies model: arbitrary alphabet case. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 460–469. Springer, Heidelberg (2006)

Turchin, V.F.: The Phenomenon of Science, a cybernetic approach to human evolution. Columbia University Press, New York (1977)

Turing, A.M.: The Chemical Basis of Morphogenesis. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237(641), 37–72 (1952)

Turney, P., Whitley, D., Anderson, R.: Evolution, Learning, and Instinct: 100 Years of the Baldwin Effect. Special Issue of Evolutionary Computation on the Baldwin Effect 4(3) (1996)

Usui, Y., Arita, T.: Situated and Embodied Evolution in Collective Evolutionary Robotics. In: Proc. of the 8th International Symposium on Artificial Life and Robotics, pp. 212–215 (2003)

Vahed, K.: The function of nuptial feeding in insects: review of empirical studies. Biological Reviews 73(1), 43–78 (1998)

Vander, A.J., Sherman, J., Luciano, D.: Human Physiology: The Mechanisms of Body Function, 5th edn. McGraw-Hill, New York (1990)

Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. IEEE Transactions on Evolutionary Computation 11(2), 151–180 (2007)

Vinter, R.: Optimal Control. Birkhauser, Boston (2000)

von Holst, E., Mittelstaedt, H.: Das reafferenzprinzip. Naturwissenschaften 37, 464–476 (1950)

von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Champaign (1966)

Waddington, C.H.: Paradigm for an evolutionary process. In: Waddington, C.H. (ed.) Towards a Theoretical Biology, vol. 2, pp. 106–128. Edinburgh University Press (1969)

Wade, U.B., Gifford, C.M.: Investigation of Power Sources for the Polar Seismic TETwalker. Tech. rept. TR 133. Center for Remote Sensing of Ice Sheets (CReSIS), University of Kansas, 2335 Irving Hill Road, Lawrence, KS 66045-7612 (November 2007)

Want, R., Hopper, A., Falco, V., Gibbons, J.: The active badge location system. ACM Transactions on Information Systems 10 (1992)

Ward, A., Jones, A., Hopper, A.: A new location technique for the active office. IEEE Personnel Communications Magazine, 42–47 (1997)

Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied Evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems 39(1), 1–18 (2002)

Webb, B.: Neural mechanisms for prediction: do insects have forward models? Trends in Neurosciences 27(5), 278–282 (2004)

Weiss, G.: Multiagent systems. A modern approach to distributed artificial intelligence. MIT Press, Cambridge (1999)

Whitacre, J.M., Pham, T.Q., Sarker, R.A.: Credit assignment in adaptive evolutionary algorithms. In: Proc. of the 8th annual conference on Genetic and evolutionary computation (GECCO 2006), ACM, New York (2006)

Whitaker, H.P.: An Adaptive System for Control of the Dynamics Performances of Aircraft and Spacecraft. Inst. Aeronautical Services (1959)

Whiten, A., Spiteri, A., Horner, V., Bonnie, K.E., Lambeth, S.P., Schapiro, S.J., de Waal, F.B.M.: Transmission of multiple traditions within and between chimpanzee groups. Current Biology 17(12), 1038–1043 (2007)

Wiener, N.: Cybernetics: or Control and Communication in the Animal and the Machine. MIT Press, Cambridge (1948)

Wilkinson, S.: Gastrobots – Benefits and Challenges of Microbial Fuel Cells in Food Powered Robot Applications. Autonomous Robots 9(2), 99–111 (2000)

Williams, G.C.: Adaptation and Natural Selection. Princeton University Press, Princeton (1996)

Williams, G.C.: Adaptation and Natural Selection. Princeton University Press, Princeton (1966)

Winfield, A.F.T., Harper, C.J., Nembrini, J.: Towards Dependable Swarms and a New Discipline of Swarm Engineering. In: Şahin, E., Spears, W. (eds.) Swarm Robotics Workshop: State-of-the-art Survey, pp. 126–142. Springer, Heidelberg (2005)

Winfield, A.F.T., Harper, C.J., Nembrini, J.: Towards the Application of Swarm Intelligence in Safety Critical Systems. In: Proc. of the 1st International Conference on System Safety, pp. 89–95. IET Press, London (2006b)

Winfield, A.F.T., Liu, W., Nembrini, J., Martinoli, A.: Modelling A Wireless Connected Swarm of Mobile Robots. Swarm Intelligence 2(2-4), 241–266 (2008)

Winfield, A.F.T., Nembrini, J.: Safety in numbers: fault-tolerance in robot swarms. International Journal of Modelling, Identification and Control 1(1), 30–37 (2006)

Wingate, D., Singh, S.P.: Kernel Predictive Linear Gaussian models for nonlinear stochastic dynamical systems. In: Proc. of the International Conference on Machine Learning (ICML 2006). ACM International Conference Proceeding Series, vol. 148, pp. 1017–1024 (2006)

Winkler, L., Wörn, H.: Symbricator3D - A Distributed Simulation Environment for Modular Robots. In: Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z. (eds.) ICIRA 2009. LNCS, vol. 5928, pp. 1266–1277. Springer, Heidelberg (2009)

Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks 11(7-8), 1317–1329 (1998)

Wolpert, L.: Positional information and the spatial pattern of cellular differentiation. Journal of Theoretical Biology 25(1), 1–47 (1969)

Wolpert, L.: Principles of Development. Oxford University Press, Oxford (1998)

Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons, Chichester (2002)

Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]. IEEE Robotics & Automation Magazine 14(1), 43–52 (2007)

Yim, M., Zhang, Y., Roufas, K., Duff, D., Eldershaw, C.: Connecting and disconnecting for chain self-reconfiguration with PolyBot. IEEE/ASME Transactions on mechatronics, special issue on Information Technology in Mechatronics (2003)

Zaslavsky, G.M.: The Physics of Chaos in Hamiltonian Systems, 2nd edn. Imperial College Press, London (2007)

Zetterström, G.: Collaborative actuation in microrobotic swarm based on collective decision making and surfacecolor identification. Master Thesis, University of Stuttgart, Germany (2006)

Zhao, Y.J., Govindan, R., Estrin, D.: Residual energy scan for monitoring sensor networks. In: Proc. of the IEEE Conference on Wireless Communications and Networking, vol. 1, pp. 356–362 (2002)

Ziemke, T.: What's that thing called Embodiment? In: Proc. of the 25th Annual meeting of the Cognitive Science Society, pp. 1305–1310 (2002)

Zimmermann, H.-J.: Fuzzy Set Theory - and its Application, 4th edn. Springer, Heidelberg (2001)

Zonia, L., Bray, D.: Swimming patterns and dynamics of simulated *Escherichia coli* bacteria (in press). Journal of the Royal Society Interface (2009)

Zweigle, O., Käppeler, U., Rajaie, H., Häußermann, K., Tamke, A., Koch, A., Eckstein, B., Aichele, F., Levi, P.: 1. RFC Stuttgart Team Description 2009. In: Graz, T.U. (ed.) RoboCup 2009 International Symposium, pp. 1–8. TU Graz, Graz (2009)

Zykov, V., Mytilinaios, E., Desnoyer, M., Lipson, H.: Evolved and Designed Self-Reproducing Modular Robotics. IEEE Transactions on Robotics 23(2), 308–319 (2007a)

Zykov, V., Chan, A., Lipson, H.: Molecubes: An Open-Source Modular Robotics Kit. In: Proc. of the International Conference on Intelligent Robots and Systems (IROS 2007), IEEE/RSJ Workshop on Self-Reconfigurable Robotics (2007b)

Zykov, V., William, P., Lassabe, N., Lipson, H.: Mechanical Molecubes Extended: Diversifying Capabilities of Open-Source Modular Robotics. In: Proc. of the International Conference on Intelligent Robots and Systems (IROS 2008), IEEE/RSJ Workshop on Self-Reconfigurable Robotics (2008)

Index

Cognitive Systems Monographs

Edited by R. Dillmann, Y. Nakamura, S. Schaal and D. Vernon

Vol. 1: Arena, P.; Patanè, L. (Eds.):
Spatial Temporal Patterns for
Action-Oriented Perception
in Roving Robots
425 p. 2009 [978-3-540-88463-7]

Vol. 2: Ivancevic, T.T.; Jovanovic, B.;
Djukic, S.; Djukic, M.; Markovic, S.:
Complex Sports Biodynamics
326 p. 2009 [978-3-540-89970-9]

Vol. 3: Magnani, L.:
Abductive Cognition
534 p. 2009 [978-3-642-03630-9]

Vol. 4: Azad, P.:
Visual Perception for Manipulation
and Imitation in Humanoid Robots
270 p. 2009 [978-3-642-04228-7]

Vol. 5: de de Aguiar, E.:
Animation and Performance Capture
Using Digitized Models
168 p. 2009 [978-3-642-10315-5]

Vol. 6: Ritter, H.; Sagerer, G.;
Dillmann, R.; Buss, M.:
Human Centered Robot Systems
216 p. 2009 [978-3-642-10402-2]

Vol. 7: Levi, P.; Kernbach, S. (Eds.):
Symbiotic Multi-Robot Organisms
467 p. 2010 [978-3-642-11691-9]