Emergence, Complexity and Computation ECC

Georgios Ch. Sirakoulis
Andrew Adamatzky *Editors*

Robots and Lattice Automata

Springer

Emergence, Complexity and Computation

Volume 13

About this Series

The Emergence, Complexity and Computation (ECC) series publishes new developments, advancements and selected topics in the fields of complexity, computation and emergence. The series focuses on all aspects of reality-based computation approaches from an interdisciplinary point of view especially from applied sciences, biology, physics, or chemistry. It presents new ideas and interdisciplinary insight on the mutual intersection of subareas of computation, complexity and emergence and its impact and limits to any computing based on physical limits (thermodynamic and quantum limits, Bremermann's limit, Seth Lloyd limits…) as well as algorithmic limits (Gödel's proof and its impact on calculation, algorithmic complexity, the Chaitin's Omega number and Kolmogorov complexity, non-traditional calculations like Turing machine process and its consequences,…) and limitations arising in artificial intelligence field. The topics are (but not limited to) membrane computing, DNA computing, immune computing, quantum computing, swarm computing, analogic computing, chaos computing and computing on the edge of chaos, computational aspects of dynamics of complex systems (systems with self-organization, multiagent systems, cellular automata, artificial life,…), emergence of complex systems and its computational aspects, and agent based computation. The main aim of this series it to discuss the above mentioned topics from an interdisciplinary point of view and present new ideas coming from mutual intersection of classical as well as modern methods of computation. Within the scope of the series are monographs, lecture notes, selected contributions from specialized conferences and workshops, special contribution from international experts.

More information about this series at http://www.springer.com/series/10624

Georgios Ch. Sirakoulis · Andrew Adamatzky
Editors

Robots and Lattice Automata

Springer

Editors
Georgios Ch. Sirakoulis
Department of Electrical and Computer Engineering
Democritus University of Thrace
Xanthi
Greece

Andrew Adamatzky
Unconventional Computing Centre
University of the West of England
Bristol
UK

ISSN 2194-7287 ISSN 2194-7295 (electronic)
ISBN 978-3-319-10923-7 ISBN 978-3-319-10924-4 (eBook)
DOI 10.1007/978-3-319-10924-4

Library of Congress Control Number: 2014951740

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Robots and Automata are notionally related. In this context, Automata (originated from the latinization of the Greek word "αυτόματον") as self-operating autonomous machines, invented from ancient years can be easily considered as the first steps of these robotic-like efforts. On other words, an Automaton is a self-operating machine, while a robot is a hardware agent with role(s) to operate usually without an immediate human operator. Automata are useful tools for formal descriptions of robots. Automata themselves are formally represented by final state machines: the abstract machines which take finite number of states and change their state while triggered by certain conditions. Authors of the book bring together concepts, architectures and implementations of Lattice Automata and Robots. Lattice Automata are minimal universal instantiation of space and time. A Lattice Automaton is either a regular array of finite state machines or collectives of mobile finite state machines inhabiting a discrete space. In both cases the finite states machines, or Automata, update their states by the same rules depending on states of their immediate neighbours. Automata and Robots often share the same notional meaning: Automata are mathematical models of robots and also they are integral parts of robotic control systems.

The book opens with inspiring text by Rosenberg—Chap. 1—on computational potential of groups of identical finite-state machines. The chapter lays somewhat foundational theoretical background for the rest of the book.

Modular robots are kinematic machines of many units capable for changing its topology by dynamically updating connections between the units. To develop efficient algorithms of reconfiguration, we represent the robotic units by configurations of Lattice Automata and study Automaton transition rules corresponding to reconfiguration. The topic is studied in full details in three chapters: Chap. 2 by Stoy introduces the reader to the theoretical and general aspects of modular reconfigurable robots in Lattice Automata; Chap. 3 by Eckenstein and Yim reproduces all the up-to-date related works and corresponding modular reconfigurable robotic systems; while in Chap. 4, Tomita and co-authors provide full details for some of these modular systems, namely Fractum and M-Tran in every possible aspect and discuss the general problems of Lattice-based robotic systems.

Motion control and path planning are amongst key problems of robotics, they put high demands on detailed knowledge of environment and consume substantial computational resources. Five chapters explicitly deal with these problems. Thus, Arena and co-authors, in Chap. 5, use Automaton networks to control locomotion of the fly-inspired robot. Efficient ways of routing, an abstract version of path planning, are designed and analysed by Hoffman and Désérable in Chap. 6. Marchese proposes to use particular families of Cellular Automata to provide an optimal representation of space and maps in precise parallel motion planning, in Chap. 7. Charalampous and co-authors in Chap. 8 adapt classical designs of Cellular Automaton based shortest path finders to undertake autonomous collision-free navigation. Moreover, Ioannidis and co-authors proposed the employment of Cellular Automata advanced with Ant Colony Optimization techniques resulting to Cellular Robotic Ants synergy coordination for tackling the path planning problem for robotic teams in Chap. 9.

Further applications of Lattice Automata in Robotics are presented in the following chapters. A novel method of map representation is proposed in Chap. 10 by Kapoutsis and his co-authors. There, a configuration of elevation heights is converted to cells' states; thus, an entire map is represented by a Cellular Automaton configuration. Cellular Automata have been a classical tool in image processing community since mid-1970s, yet, there is still vast lands of unexplored features and algorithms. In his Chap. 11, Nalpantidis demonstrates practical, real-life implementation of Cellular Automaton algorithms onboard of a mobile robot.

The last two chapters deal with cooperative actions in large-scale robotic collectives. In both chapters, robots are oscillating mechanisms arranged on a two-dimensional array: their aim is to adjust their oscillations or states to produce a specified vibration pattern. Silva and co-authors, in Chap. 12 provide modelling and analysis of the space–time behaviour of such collectives and transitions between different modes of behaviour. Application of the vibrating automaton array to physical manipulator of objects in real life is studied by Georgilas and co-authors in Chap. 13. They show how Automaton model of an sub-excitable medium can be used to purposefully transport objects.

All chapters are written in an accessible manner and lavishly illustrated. The book will help computer and robotic scientists and engineers to understand mechanisms of decentralised functioning of robotic collectives and to design future and emergent reconfigurable, parallel and distributed robotic systems.

Georgios Ch. Sirakoulis
Andrew Adamatzky

Contents

Chapter 1
Algorithmic Insights into Finite-State Robots

Arnold L. Rosenberg

Abstract Modern technology has enabled the deployment of small computers that can act as the "brains" of mobile robots. Multiple advantages accrue if one can deploy simpler computers rather than more sophisticated ones: For a fixed cost, one can deploy more computers, hence benefit from more concurrent computing and/or more fault-tolerant design—both major issues with assemblages of mobile "intelligent" robots. This chapter explores the capabilities and limitations of computers that execute simply structured *finite-state programs*. The robots of interest operate within constrained physical settings such as warehouses or laboratories; they operate on tesselated "floors" within such settings—which we view formally as meshes of tiles. The major message of the chapter is that teams of (identical) robots whose "intellects" are powered by finite-state programs are capable of more sophisticated algorithmics than one might expect, even when the robots must operate: (*a*) *without the aid of centralized control* and (*b*) using algorithms that are *scalable*, in the sense that they work in meshes/"floors" of arbitrary sizes. A significant enabler of robots' algorithmic sophistication is their ability to use their host mesh's edges—i.e., the walls of the warehouses or laboratories—when orchestrating their activities. The capabilities of our "finite-state robots" are illustrated via a variety of algorithmic problems that involve path planning and exploration, in addition to the rearranging of labeled objects.

1.1 Introduction

Modern technology has enabled the deployment of small computers that can act as the "brains" of mobile robots. Multiple advantages accrue if one can deploy simpler computers rather than more sophisticated ones: For a fixed cost, one can deploy more computers, hence benefit from more concurrent computing and/or more fault-tolerant design—both major issues with assemblages of mobile "intelligent" robots. This chapter explores the capabilities and limitations of computers that execute simply

A.L. Rosenberg (✉)
Northeastern University, Boston, MA 02115, USA
e-mail: rsnbrg@ccs.neu.edu

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_1

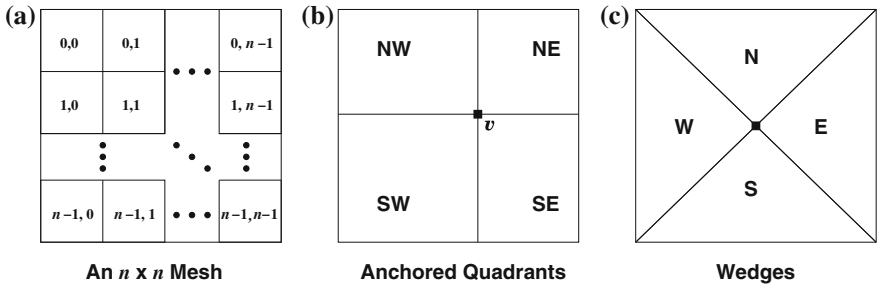


Fig. 1.1 **a** The $n \times n$ mesh $\mathcal{M}_n$; **b** $\mathcal{M}_n$ partitioned into the four *quadrants* determined by *anchor tile* v. **c** $\mathcal{M}_n$ partitioned into its four *wedges*

structured *finite-state programs*, perhaps the simplest type of program that one can expect to enable sophisticated robot behavior. The robots of interest—which we call *finite-state machines* (*FSMs*, for short), to emphasize the finite-state property—operate within constrained physical settings such as warehouses or laboratories; they operate on tesselated "floors" within such settings—which we view formally as instances of the $n \times n$ *mesh of tiles* $\mathcal{M}_n$ (Fig. 1.1a).

The major message of the chapter is that teams of (identical) FSMs are capable of more sophisticated algorithmics than one might expect, even when the FSMs must operate: (*a*) *without the aid of centralized control* and (*b*) using algorithms that are *scalable*, in the sense that they work in meshes/"floors" of arbitrary sizes. A significant enabler of robots' algorithmic sophistication is their ability to exploit the *edges* of the meshes they operate in—i.e., the walls of the warehouses or laboratories—when orchestrating their activities. The capabilities of finite-state robots are illustrated here via a variety of algorithmic problems that involve path planning and exploration, in addition to the rearranging of labeled objects (that sit within some tiles of the home mesh). We note again the major points that FSMs operate *without centralized control* while executing algorithms that are *scalable*.

Our study focuses on algorithmic problems that emerge from complementary avenues of investigation with histories that span several decades. The literature on automata theory and its applications contains studies such as [3, 5, 8, 10, 24, 27] that focus on the (in)ability of FSMs to explore graphs with goals such as finding "entrance"-to-"exit" paths or exhaustively visiting all of a graph's nodes or all of its edges. Other studies, e.g., [4, 15, 17, 23, 26, 36], focus on algorithms that enable FSMs that populate the tiles of (multidimensional) meshes—*cellular automata*—to tightly synchronize, a crucial component of many activities that must be performed without centralized control; the cellular automaton model dates back a half-century [38] but remains of interest today [14, 39]. Yet other automata-theoretic studies update the historical string-recognition work of classical finite-automata theory—cf., [25, 28]—to more ambitious domains such as graphics [7, 13, 19, 20, 29]. The robotics literature contains numerous studies—e.g., [1, 2, 11, 18, 35]—that explore ants as a metaphor for simple robots that collaborate to accomplish complex tasks;

the interesting topic of "virtual pheromones" within this metaphor is studied in [11, 18, 31, 35]. Cellular automata appear in many studies of robotic applications of automata-theoretic concepts: application- and implementation-oriented studies as well as theoretical ones [6, 11, 16, 22, 32, 35, 37]. The current chapter melds the automata-theoretic and robotic points of view by studying FSMs that operate within square meshes; most of the problems we discuss are more closely motivated by robotics than automata theory, although a few emerge from the world of language-oriented studies.

The specific algorithmic challenges that we study are inspired by our earlier work on FSMs, which itself emerged from our work on the Cellular ANTomaton model [32], a marriage of robotics and cellular automata. All of our studies demand algorithms that are *scalable* in the sense that they work in meshes $\mathcal{M}_n$ of arbitrary size, i.e., for arbitrarily large values of n. Our first study involving FSMs was [31], which focused on the *Parking Problem* for FSMs; this problem requires each FSM in a mesh to go to its closest corner and has FSMs within each corner organize into a maximally compact formation (i.e., one that minimizes the FSMs' aggregate distance to their nearest corners). A central component of parking is to have each FSM determine which *quadrant* of $\mathcal{M}_n$ it resides in (cf. Fig. 1.1b); because the home-quadrant determination problem is treated in detail in [31], we focus here on a kindred, but rather different problem that requires FSMs to determine their home *wedges* (cf. Fig. 1.1c). Our next study of FSMs, in [33], allowed the tiles of $\mathcal{M}_n$ to be labeled from a given repertoire. The study required FSMs to move to a specified tile $v_{\varphi,\psi} = \langle \lfloor \varphi n \rfloor, \lfloor \psi n \rfloor \rangle$ of $\mathcal{M}_n$, identified by a prespecified pair of positive rational numbers $\langle \varphi, \psi \rangle$.

> Note that: (a) the rational numbers $\varphi = a/b$ and $\psi = c/d$ are *fixed for each specific problem-instance*, and (b) the FSM $\mathcal{F}$ that solves each instance of the problem is designed so that its state-memory "contains" the four integers a, b, c, d; i.e., $\mathcal{F} = \mathcal{F}^{(a,b,c,d)}$. The scalability in our problem solutions refers only to the mesh-size paramater n.

Every tile $v_{\varphi,\psi}$ can serve as an *anchor* to induce a partition of $\mathcal{M}_n$ into quadrants, as in Fig. 1.1b. The added challenge in [33] is to have the FSMs *sweep* the quadrants induced by $v_{\varphi,\psi}$ to check that each of the mesh's tiles contains a quadrant-specific label. In the third of our studies, [34], FSMs do more than plan application-specific trajectories and seek specified goal-tiles. They now rearrange objects that occupy $\mathcal{M}_n$'s tiles in various prespecified ways while transporting the objects from $\mathcal{M}_n$'s top row to its bottom row. The specific rearrangments include: (1) reversing the order in which objects appear, (2) cyclically rotating the objects, and (3) sorting the objects by their (ordered) "types." In addition to being scalable, the algorithms we describe in [34] are *pipelineable* in a way that achieves parallel speedup that is asymptotically linear in the number of FSMs (even as that number approaches n). The pipelining that we refer to here has a team of identical copies of an FSM $\mathcal{F}$ march one after the other, performing different instances of the chores to be performed; see, e.g., [34] for details.

The problems we discuss in this chapter describe, and in several places extend or improve, the material in [32–34]. The problems we discuss require FSMs:

- to determine where they are within $\mathcal{M}_n$;
  We focus on having FSMs determine which *wedge* of $\mathcal{M}_n$ they reside in (cf. Fig. 1.1c). (Recall that we treat the analogous problem for quadrants in [31].)
- to seek various target tiles of $\mathcal{M}_n$;
  We recapitulate the study in [33], wherein target tiles are specified via pairs of positive rational numbers, specifically using the rational pair $\langle \varphi, \psi \rangle$ to specify tile $\langle \lfloor \varphi(n-1) \rfloor, \lfloor \psi(n-1) \rfloor \rangle$ of $\mathcal{M}_n$.
- to transport the objects residing in $\mathcal{M}_n$'s top row to $\mathcal{M}_n$'s bottom row while rearranging the objects in prespecified ways;
  Excerpting from our study in [34], we have FSMs (1) reverse the objects' original order, (2) cyclically rotate the original order, and (3) sort the objects by their (ordered) types.
- to determine whether the objects residing in certain of $\mathcal{M}_n$'s rows of tiles have certain patterns.
  We complement the study in [34] by having FSMs check the pattern of objects along $\mathcal{M}_n$'s rows rather than effect the pattern. We have FSMs identify *palindromes* (words that read the same forwards and backwards), *perfect squares* (even-length words whose first and second halves are identical), and *rotations* (a pair of words one of which is a cyclic rotation of the other).

The algorithmic tools employed by our FSMs extend to myriad other problems.

A final word of introduction. We noted earlier that various sources—e.g., [11, 18, 35]—discuss "virtual pheromones" as a control mechanism for robotic "ants." This mechanism assigns registers within each robot's internal computer to maintain levels of intensity of an array of pheromones, thereby implementing a digital analogue of the volatile organic compounds that are used by nature's ants. We largely ignore "virtual pheromones" because FSMs do not need them to execute the algorithms we discuss. We note only that in [31] we have shown that "virtual pheromones" do not enhance the power of a single FSM—although they can sometimes be used to decrease the required size of an FSM, as measured in number of states.

1.2 Technical Background

1.2.1 FSM "Robots" and Their Domains

Our formal model of *FSM-robots* (*FSM*s, for short) is obtained by augmenting the capabilities of standard finite-state machines (sources such as [30] provide formal details) with the ability to travel around square *meshes* of *tiles*, possibly transporting *objects* from one tile to another (empty) one. We flesh out this informal description.

Meshes. We index the n^2 tiles of the $n \times n$ mesh $\mathcal{M}_n$ by the set[1] $[0, n-1] \times [0, n-1]$; see Fig. 1.1a. The set of tiles of $\mathcal{M}_n$ that share the first index-coordinate i, i.e., $R_i \stackrel{\text{def}}{=} \{\langle i, j\rangle \mid j \in [0, n-1]\}$, is the ith *row* of $\mathcal{M}_n$; the set of tiles that share the second index-coordinate j, i.e., $C_j \stackrel{\text{def}}{=} \{\langle i, j\rangle \mid i \in [0, n-1]\}$, is the jth *column* of $\mathcal{M}_n$. Tile $\langle i, j\rangle$ of $\mathcal{M}_n$ is:

- a *corner* tile if $i, j \in \{0, n-1\}$;
- an *(internal) edge* tile if it is one of:
  - a *bottom* tile Meaning that $i = 0$ and $j \in [1, n-2]$;
  - a *top* tile Meaning that $i = n-1$ and $j \in [1, n-2]$;
  - a *left* tile Meaning that $i \in [1, n-2]$ and $j = 0$;
  - a *right* tile Meaning that $i \in [1, n-2]$ and $j = n-1$;
- an *internal* tile if $i, j \in [1, n-2]$.

We employ the *King's move* adjacency model for meshes, so named for the chess piece (also known as the Moore model). Under this model, each tile $\langle i, j\rangle$ of $\mathcal{M}_n$ has up to 8 neighbors, one in each compass direction, abbreviated (in clockwise order) N, NE, E, SE, S, SW, W, NW. Accordingly, each internal tile of $\mathcal{M}_n$ has 8 neighbors; each (internal) edge tile has 5 neighbors; and each corner tile has 3 neighbors. Clerical modifications allow any *fixed finite* set of adjacencies, each specified by a pair of signed positive integers $\langle \pm a, \pm b\rangle$; each such pair, $\langle c, d\rangle$, indicates that every tile $\langle i, j\rangle$ of $\mathcal{M}_n$ has a neighbor at index-point $\langle i + c, j + d\rangle$, *as long as this point is a valid index for* $\mathcal{M}_n$, meaning that both $i + c$ and $j + d$ are in the range $[0, n-1]$. One opts for program compactness at the cost of algorithmic efficiency by choosing a smaller repertoire of adjacencies, such as *NEWS* moves: N, E, W, S (which are also known as the von Neumann model); one opts for increased efficiency at the cost of larger programs by choosing a larger repertoire of adjacencies, such as the 16 *Knight's + King's* moves. These three alternatives are illustrated in Fig. 1.2, which depicts the world from the viewpoint of an FSM. Whichever adjacency model is implemented: *every edge of every tile v of $\mathcal{M}_n$ is labeled to indicate which of v's potential neighbors actually exist.* (This enables FSMs to avoid "falling off" $\mathcal{M}_n$ or "banging into a wall.")

$\mathcal{M}_n$'s four *quadrants* are determined by lines that cross at an *anchor* tile v and are perpendicular to $\mathcal{M}_n$'s edges (Fig. 1.1b). The "standard" quadrants—which are anchored at $\mathcal{M}_n$'s "center" tile $\langle \lfloor \frac{1}{2}(n-1) \rfloor, \lfloor \frac{1}{2}(n-1) \rfloor \rangle$, hence are as close to equal in number of tiles as the parity of n allows—comprise the following sets of tiles.

[1] For positive integers i and $j > i$, we denote by $[i, j]$ the set $\{i, i+1, \ldots, j\}$.

Quadrant	Name	Tile-set
SOUTHWEST	$\mathcal{Q}_{SW}$	$\{\langle x, y\rangle \mid x \geq \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$
NORTHWEST	$\mathcal{Q}_{NW}$	$\{\langle x, y\rangle \mid x < \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$
SOUTHEAST	$\mathcal{Q}_{SE}$	$\{\langle x, y\rangle \mid x \geq \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y < \lfloor \frac{1}{2}(n-1)\rfloor\}$
NORTHEAST	$\mathcal{Q}_{NE}$	$\{\langle x, y\rangle \mid x < \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$

$\mathcal{M}_n$'s four *wedges* are determined by passing lines with slopes ± 1 through $\mathcal{M}_n$'s "center" tile; see Fig. 1.1c. These lines come as close to connecting $\mathcal{M}_n$'s corners as the parity of n allows. $\mathcal{M}_n$'s wedges comprise the following sets of tiles.

Wedge	Name	Tile-set
NORTH	$\mathcal{W}_N$	$\{\langle x, y\rangle \mid [x \leq y] \text{ and } [x + y \leq n - 1]\}$
SOUTH	$\mathcal{W}_S$	$\{\langle x, y\rangle \mid [x > y] \text{ and } [x + y \geq n]\}$
EAST	$\mathcal{W}_E$	$\{\langle x, y\rangle \mid [x \leq y] \text{ and } [x + y \geq n]\}$
WEST	$\mathcal{W}_W$	$\{\langle x, y\rangle \mid [x > y] \text{ and } [x + y \leq n - 1]\}$

Rounding ensures that each tile has a unique home quadrant and home wedge.

Objects. Each tile v of $\mathcal{M}_n$ can be empty—i.e., v contains 0 FSMs and 0 objects—or it can hold at most one FSM and at most one object—i.e., v contains 0 FSMs and 1 object *or* 1 FSM and 0 objects *or* 1 FSM and 1 object. Each object has a *type* chosen from some *fixed finite ordered* set. Because the number of objects can be commensurate with n while the number of object-types must be fixed independent of n, perforce, many objects can have the same type.

FSMs. At any moment, an FSM $\mathcal{F}$ occupies a single tile of $\mathcal{M}_n$, possibly sharing that tile with an object but *not* with another FSM. At each step, $\mathcal{F}$ can move to any neighbor v' of its current tile v (cf. Fig. 1.2), providing that v' contains no other FSM. Additionally, if v' contains no object, the $\mathcal{F}$ can convey the object that resides on v (if there is one) to v'.

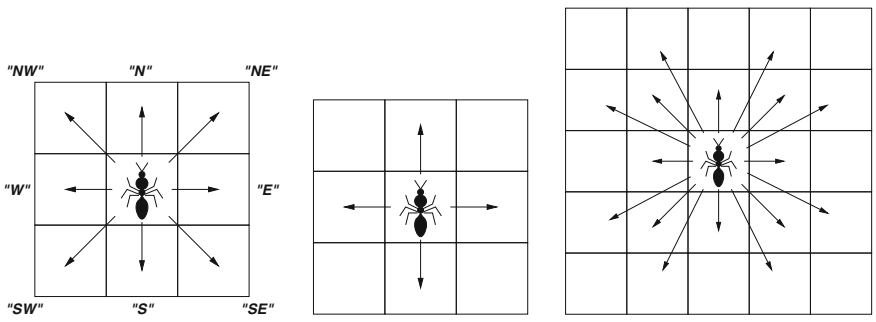


Fig. 1.2 Single-step move repertoires for FSMs. (*left*) The *King's-move* repertoire; (*center*) the *NEWS* (North-East-West-South) repertoire; (*right*) the *Knight's-move* + *King's-move* repertoire

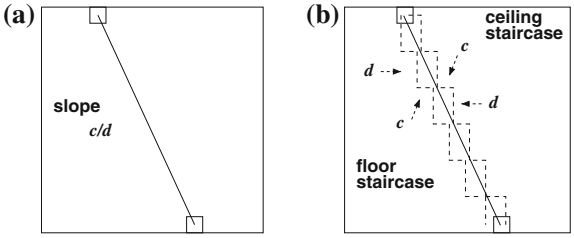


Fig. 1.3 An idealized "smooth" straight path with slope c/d depicted by *solid lines*. The smooth path appears alone in subfigure (**a**); in subfigure (**b**), it is accompanied by its two discretized *staircase* versions, depicted by *dashed lines*. Both staircases alternate horizontal subpaths of length c with vertical subpaths of length d. If the smooth path begins at tile $\langle 0, i\rangle$ along $\mathcal{M}_n$'s *top edge*, then the *floor* staircase leads to tile $\langle n-1, i + \lfloor (c/d)(n-1)\rfloor\rangle$, and the *ceiling* staircase leads to tile $\langle n-1, i + \lceil (c/d)(n-1)\rceil\rangle$

FSM trajectories. As $\mathcal{F}$ plans its next move, it *must* consider the label of its current tile—hence be aware of residing on an edge or corner tile of $\mathcal{M}_n$ (in order to avoid "falling off" the mesh or "banging into its walls"). But, being an FSM, $\mathcal{F}$ *cannot* exploit any knowledge of the size-parameter n of the mesh it resides in—except for "finite-state" knowledge such as the parity of n.

Our algorithms often mandate that an FSM $\mathcal{F}$ traverse straight paths with various (rational) slopes within $\mathcal{M}_n$. We talk informally as though $\mathcal{F}$ can traverse a path of arbitrary slope $\varphi = c/d$, as suggested in Fig. 1.3a. However, no matter what single-step move repertoire one enables FSMs to employ, for most rational slopes φ, $\mathcal{F}$ can only *approximate* following a path of slope φ. We have $\mathcal{F}$ accomplish such an approximation by discretizing the path of slope φ, in the manner depicted in Fig. 1.3b, via either a *floor* staircase or a *ceiling* staircase. The choice between the two types of staircase depends on whether $\mathcal{F}$ gets a better solution to the problem that it is solving from undershooting or overshooting the ideal target tile in $\mathcal{M}_n$'s bottom row. Both discretizing staircases approximate the ideal slope $\varphi = c/d$ of Fig. 1.3a via a *staircase* whose stairs each comprise alternating d vertical steps and c horizontal steps. (This strategy works also for the smaller repertoire of NEWS adjacencies.) The floor staircase begins with a vertical subpath, while the ceiling staircase begins with a horizontal subpath. Focus on an ideal path of slope c/d that begins at tile $\langle 0, i\rangle$ in $\mathcal{M}_n$'s top row. Its discretizing staircases also begin at this tile. Depending on the value of $n \bmod d$, the discretizing staircases may "miss"—i.e., not terminate at— the ideal target tile $\langle n-1,\ i + (c/d)(n-1)\rangle$ in $\mathcal{M}_n$'s bottom row. In this case, a finalizing partial stair must be added to the staircase in order to complete the path to the target tile. $\mathcal{F}$ recognizes the need for the final stair when it attempts to move d steps downward but encounters $\mathcal{M}_n$'s bottom edge after only $d' < d$ steps. In response, $\mathcal{F}$ terminates its downward journey and instead moves some number $\delta(d') \leq c$ of steps rightward (for the floor staircase) or leftward (for the ceiling staircase). After the final correction, $\mathcal{F}$ ends up in tile $\langle n-1,\ i + \lfloor (c/d)(n-1)\rfloor\rangle$ via the floor staircase or tile $\langle n-1,\ i + \lceil (c/d)(n-1)\rceil\rangle$ via the ceiling staircase. The offset $\delta(d')$ depends only on c and $d' = n \bmod d$; it does not depend on i or on the structure of $\mathcal{F}$. The computation of $\delta(d')$ from d' can be stored in a table within $\mathcal{F}$'s states, using no more than $d \log_2 c$ bits of storage. To summarize:

$$\begin{array}{l}\text{The floor staircase takes } \mathcal{F} \text{ from } \langle 0, i\rangle \text{ to } \langle n-1,\ i + \lfloor (c/d)(n-1)\rfloor\rangle;\\ \text{The ceiling staircase takes } \mathcal{F} \text{ from } \langle 0, i\rangle \text{ to } \langle n-1,\ i + \lceil (c/d)(n-1)\rceil\rangle.\end{array} \tag{1.1}$$

Multiple FSMs on $\mathcal{M}_n$. A team of FSMs on $\mathcal{M}_n$ can be activated (from the outside world) simultaneously. (One can often use the Firing Squad Synchronization algorithm [15] to achieve this simultaneity.) Distinct FSMs on $\mathcal{M}_n$ operate synchronously, i.e., can follow trajectories *in lockstep*. FSMs that reside on neighboring tiles are aware of each other and can pass bounded-length messages—such as "I AM HERE" or "I WANT TO MOVE TO YOUR TILE." Such simple messages often enable one FSM to act as an "usher"/"shepherd" for other FSMs. Each FSM's moves on $\mathcal{M}_n$ are tightly orchestrated. Specifically, an FSM attempts to move in direction:

N Only at steps $t \equiv 0 \bmod 8$;	NE Only at steps $t \equiv 1 \bmod 8$;
E Only at steps $t \equiv 2 \bmod 8$;	SE Only at steps $t \equiv 3 \bmod 8$;
S Only at steps $t \equiv 4 \bmod 8$;	SW Only at steps $t \equiv 5 \bmod 8$;
W Only at steps $t \equiv 6 \bmod 8$;	NW Only at steps $t \equiv 7 \bmod 8$;

(A repertoire of k atomic moves would require a modulus of k.) This orchestration means that *FSMs need never collide!* If several FSMs want to enter a tile v from (perforce distinct) neighboring tiles, then one will have permission to move before the others—so all FSMs will learn about the conflict before a collision occurs.

Because all FSMs are identical, pairs of FSMs can simulate the effect of crossing over one another by "exchanging roles." Say, for instance, that FSM $\mathcal{F}_1$ resides at some tile $\langle i, j\rangle$ while FSM $\mathcal{F}_2$ resides at a neighboring tile. If $\mathcal{F}_1$ wants to proceed through $\mathcal{F}_2$'s tile, then the two FSMs can simulate the crossover by exchanging identities. Of course, this can be an expensive operation, since $\mathcal{F}_1$ and $\mathcal{F}_2$ must exchange state information in order to implement this simulation strategy. Consequently, it is usually advantageous to modify the FSMs' programs, if possible, either to avoid such crossovers or to have batches of them happen simultaneously. In principle, though, such exchanges add only $O(1)$ cost to the FSMs' times.

1.2.2 Algorithmic Standards and Simplifications

*Algorithms are **finite-state***. Each is specified by a single *finite-state program* which all robots execute. Such programs, as described in [30] and employed in "finite-state" programming systems such as *CARPET* [37], have the form depicted in Fig. 1.4. Note in the figure how statement labels play the role of states. Note also that *all FSMs are identical;* none has a "name" that renders it unique.

*Algorithms are **scalable***: *They work on meshes of arbitrary sizes.* In particular, an FSM $\mathcal{F}$ cannot exploit information about the size of a mesh $\mathcal{M}_n$, other than

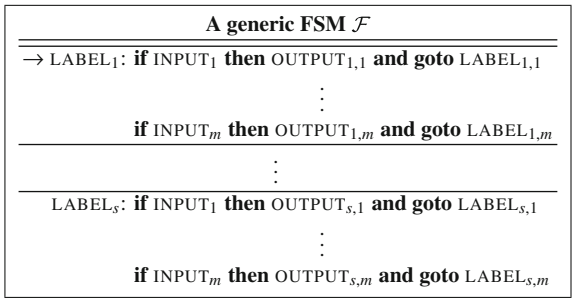


Fig. 1.4 The structure of a program for a generic FSM $\mathcal{F}$. The *start state* is indicated by an *arrow*

"finite-state" properties of n such as its parity.

Algorithms are ***decentralized*** *but* ***synchronous***. Once started, FSMs operate autonomously, but their independent clocks tick at the same rate—so that distinct FSMs can follow trajectories *in lockstep*. This assumption is no less realistic than the analogous assumption with synchronous-start human endeavors.

These guidelines are often violated in implementations, as in [9, 11, 16, 18, 35], where practical simplicity overshadows algorithmic simplicity.

We specify algorithms in English, trying to tailor the amount of detail to the complexity of the specification. Our goal is to make it clear how to craft a realizing finite-state program; if we can achieve this, then the details of our algorithm-specification language are irrelevant. To aid the reader's intuition, though, we do present one explicit "pseudo-code" FSM program, for the simple, yet nontrivial, pattern-reversal problem, in Fig. 1.10. Several other "pseudo-code" FSM programs can be found in our earlier work on FSM robots [31, 33, 34].

1.3 $\mathcal{M}_n$'s "Walls" as an Algorithmic Tool

This section is devoted to several problems that FSMs can solve efficiently by using $\mathcal{M}_n$'s edges to algorithmic advantage. We see that these edges sometimes enable FSMs on $\mathcal{M}_n$ to exploit the value of n without really "knowing" the value.

1.3.1 Algorithms Based on "Bouncing Off" $\mathcal{M}_n$'s "Walls"

For this section, we posit that each tile along designated rows of $\mathcal{M}_n$ contains an object that has a *type* from a fixed finite set; we thereby view each such row as containing a sequence of the form $\sigma_0 \ldots \sigma_{n-1}$, where each σ_i is an object-type. In Sect. 1.3.1.1 we illustrate how an FSM $\mathcal{F}$ on $\mathcal{M}_n$ can sometimes make inferences about the value of n from certain edge-to-edge walks within the mesh. In subsequent subsections, we illustrate how these inferences allow $\mathcal{F}$ to solve some rather

sophisticated problems that relate to patterns within the sequences of object-types within $\mathcal{M}_n$. The problems in Sect. 1.3.1.2 require $\mathcal{F}$ to check the pattern of object-types along $\mathcal{M}_n$'s top row; the problems in Sect. 1.3.1.3 require $\mathcal{F}$ to transport the objects along $\mathcal{M}_n$'s top row to $\mathcal{M}_n$'s bottom row, where $\mathcal{F}$ deposits the objects while rearranging the order of their types.

1.3.1.1 An Enabling Phenomenon

Let FSM $\mathcal{F}$ initiate walks whose slopes are $\pm 45°$, beginning at a tile along one of $\mathcal{M}_n$'s edges and ending at a tile along another edge. For simplicity, we focus only on walks between $\mathcal{M}_n$'s top edge and its side edges. Clerical adjustments allow one to include $\mathcal{M}_n$'s bottom edge as a source or destination. These walks can be used to *replicate* or *complement* distances along $\mathcal{M}_n$'s edges. To expand on this claim, focus on walks that begin at a tile on $\mathcal{M}_n$'s top row. Inductively, say that $\mathcal{F}$ resides on tile $v = \langle i, j \rangle$ at this moment. If $j = n - 1$, meaning that v is on $\mathcal{M}_n$'s right column, then $\mathcal{F}$ cannot take a *southeasterly* step without "falling off" $\mathcal{M}_n$; if $j < n - 1$, then $\mathcal{F}$ can take a *southeasterly* step, and a single such step moves $\mathcal{F}$ to tile $\langle i + 1, j + 1 \rangle$ (whose coordinates are obtained by adding $\langle +1, +1 \rangle$ to v's coordinates). By similar reasoning, if $j = 0$, meaning that v is on $\mathcal{M}_n$'s left column, then $\mathcal{F}$ cannot take a *southwesterly* step without "falling off" $\mathcal{M}_n$; if $j > 0$, then $\mathcal{F}$ can take a *southwesterly* step, and a single such step moves $\mathcal{F}$ to tile $\langle i + 1, j - 1 \rangle$ (whose coordinates are obtained by adding $\langle +1, -1 \rangle$ to v's coordinates). Therefore, referring to Fig. 1.5, if we focus on any integer $r \leq n/2$:

- if $\mathcal{F}$ begins at a tile $A = \langle 0, r \rangle$, then its *southwesterly* walk ends at tile $B = \langle r, 0 \rangle$ within $\mathcal{M}_n$'s left column, and its *southeasterly* walk ends at tile $C = \langle n-r-1, n-1 \rangle$ within $\mathcal{M}_n$'s right column.
- if $\mathcal{F}$ begins at a tile $D = \langle 0, n - r - 1 \rangle$, then its *southwesterly* walk ends at tile $E = \langle n - r - 1, 0 \rangle$ within $\mathcal{M}_n$'s left column, and its *southeasterly* walk ends at tile $F = \langle r, n - 1 \rangle$ within $\mathcal{M}_n$'s right column.

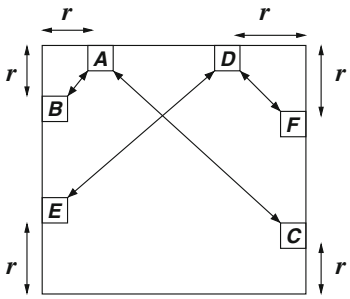


Fig. 1.5 Trajectories that lead an FSM to the *mirrors* of the tile it begins on. The point is that when the slopes of all indicated trajectories are (multiples of) 45°, then the indicated distance equalities hold (as elaborated in the text)

1.3.1.2 An FSM Identifies/Checks Patterns Along $\mathcal{M}_n$'s Top Row

We discuss two problems that each has FSM $\mathcal{F}$ verify that the sequence of object-types along $\mathcal{M}_n$'s top row has a prespecified form. An ordinary FSM cannot solve either of the illustrated problems (cf. [30]): neither corresponds to a *regular* language. In contrast, both problems yield to simple solutions that build upon the property exposed in Sect. 1.3.1.1. The problems we discuss will be familiar to any student of formal langage theory.

A. The palindrome recognition problem. This problem requires FSM $\mathcal{F}$ to determine whether the pattern of object-types, $\sigma_0\sigma_1 \dots \sigma_{n-1}$ along $\mathcal{M}_n$'s top edge is a *palindrome*—i.e., a string that reads the same forwards and backwards, so that $\sigma_0 = \sigma_{n-1},\ \sigma_1 = \sigma_{n-2}$, and so forth.

$\mathcal{F}$ begins the palindrome-recognition process by proceeding to tile $\langle 0, 0\rangle$. Inductively, when $\mathcal{F}$ is at tile $v_i = \langle 0, i\rangle$, where $i < n/2$, it remembers object-type σ_i and:

1. follows a 45° southwesterly path to tile $v_i' = \langle i, 0\rangle$ (via King's-move steps);
2. follows the horizontal path from v_i' to tile $v_i'' = \langle i, n-1\rangle$;
3. follows a 45° northwesterly path to tile $v_i''' = \langle 0, n-i\rangle$ (via King's-move steps);
4. checks whether the object-type $\sigma_{i+n/2}$ at tile v_i''' equals σ_i;
5. retraces its steps to tile v_i and moves one step rightward to tile $\langle 0, i+1\rangle$.

By horizontally iterating this process in the manner suggested in Fig. 1.6, $\mathcal{F}$ solves the palindrom-recognition problem. The correctness of the illustrated algorithm derives from Fig. 1.5 which guarantees that $\mathcal{F}$ is checking the type-equalities of the correct pairs of objects.

One can program $\mathcal{F}$ to stop in the middle of $\mathcal{M}_n$'s top edge via some sort of marking or displacement process. For instance, $\mathcal{F}$ could move each checked symbol from row 0 of $\mathcal{M}_n$ to row 1 and then halt when it encounters a symbol that has been moved; alternatively, $\mathcal{F}$ could mark each checked symbol in some way, replacing a symbol σ by an identifiable encoded version $\widehat{\sigma}$. One final pass of length n could then undo editorial marks or movements that $\mathcal{F}$ leaves during the checking process.

B. The perfect-square recognition problem. This problem requires $\mathcal{F}$ to determine whether the (even-length) pattern of object-types along $\mathcal{M}_n$'s top row,

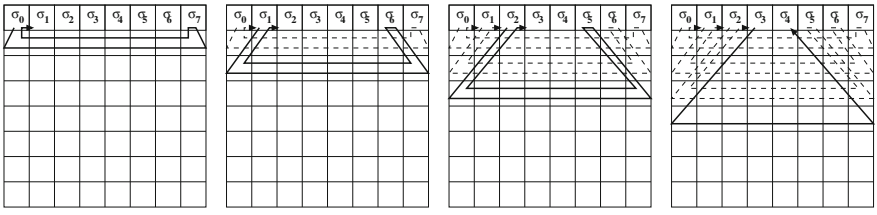


Fig. 1.6 Illustrating palindrome-checking in action. All trajectory slopes are multiples of 45°

$$\sigma_0 \ldots \sigma_{n/2-1}\sigma_{n/2} \ldots \sigma_{n-1},$$

is a *perfect square*—meaning that

$$\sigma_0 \ldots \sigma_{n/2-1} \;=\; \sigma_{n/2} \ldots \sigma_{n-1}.$$

The term "perfect square" here emerges from viewing the string of object-types as a product within the *free semigroup* [21] defined on the string's constituent symbols (which are the object-types).

We describe this problem only for the case when n is even. There are several ways to extend the problem to the case when n is odd. One of simplest is to view the middle object-type in the sequence along $\mathcal{M}_n$'s top row as a marker and have $\mathcal{F}$ verify that the sequence preceding the marker matches the sequence following the marker. This extended problem yields to a solution that is a simple modification of the one we now present for the case when n is even. (Easily, a single round-trip pass along row 0 enables $\mathcal{F}$ to determine the parity of n.)

$\mathcal{F}$ begins the perfect-square recognition process by proceeding to tile $\langle 0, 0\rangle$. Inductively, when $\mathcal{F}$ is at tile $v_i = \langle 0, i\rangle$, where $i < n/2$, it remembers object-type σ_i and:

1. follows the ceiling staircase of Fig. 1.3(b) with slope $c/d = 1/2$ to tile $v_i' = \langle n-1, i + n/2\rangle$ (recall that n is even);
2. follows the vertical path from v_i' to tile $v_i'' = \langle 0, i + n/2\rangle$;
3. checks whether the object-type $\sigma_{i+n/2}$ at tile v_i'' equals σ_i;
4. retraces its steps to tile v_i and moves one step rightward to tile $\langle 0, i + 1\rangle$.

By horizontally iterating this process in the manner suggested in Fig. 1.7, $\mathcal{F}$ solves the perfect-square recognition problem. The correctness of the illustrated algorithm derives from our derivation in Eq. 1.1 of the destination tile when $\mathcal{F}$ follows the ceiling staircase approximation to the path of slope $1/2$.

$\mathcal{F}$ terminates its processing after checking $\sigma_{n/2-1}$ against σ_{n-1}; it recognizes this moment because the latter is the type of the object in the rightmost tile of $\mathcal{M}_n$'s top row, i.e., resides along $\mathcal{M}_n$'s right edge.

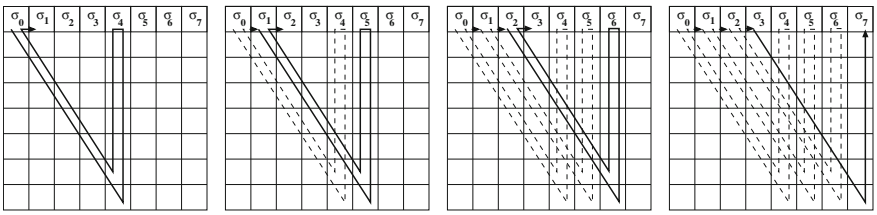


Fig. 1.7 Illustrating perfect-square checking in action. Each stage of the algorithm has $\mathcal{F}$ traverse the slope-$1/2$ ceiling staircase from a tile $\langle 0, i\rangle$ to tile $\langle n-1, i + n/2\rangle$, then follow a vertical path to $\langle 0, i + n/2\rangle$, finally checking whether the two *top-edge* symbols encountered are equal

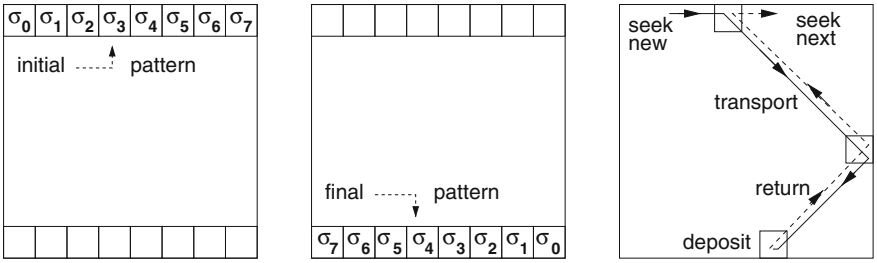


Fig. 1.8 (*left*, *center*) The initial and final configurations under the reversal rearrangement. (*right*) A generic trajectory that effects the reversal rearrangement. $\mathcal{F}$ follows the solid path as transports the current object from its initial *tile along row* 0 of $\mathcal{M}_n$ (the *top row*) to its destination *tile along row* $n-1$ (the *bottom row*). $\mathcal{F}$ then follows the *dashed path* as it returns to $\mathcal{M}_n$'s *top row* to get the next object

1.3.1.3 An FSM Rearranges Patterns from $\mathcal{M}_n$'s Top Row to Its Bottom Row

For this section, we again posit that the tiles along the top row of $\mathcal{M}_n$ each contains an object that has a *type* from a fixed finite repertoire. We describe two problems that each has a single FSM $\mathcal{F}$ transport the objects from $\mathcal{M}_n$'s top row to $\mathcal{M}_n$'s bottom row, depositing the objects according to a designated rearranged pattern of their types.

A. The reversal rearrangement. The *pattern-reversal* problem has $\mathcal{F}$ transport the objects from $\mathcal{M}_n$'s top row to $\mathcal{M}_n$'s bottom row, where it deposits them so as to reverse the pattern of their object-types; cf. Fig. 1.8(left, center). Figure 1.9 illustrates one way in which $\mathcal{F}$ can accomplish this operation on $\mathcal{M}_8$, using the "algorithmic template" of Fig. 1.8(right). In brief, the algorithm proceeds as follows. For each object σ along row 0, in left-to-right order, $\mathcal{F}$ follows a (southeasterly) trajectory of slope $-45°$ until it encounters column $n-1$ ($\mathcal{M}_n$'s right edge), and thence follows a (southwesterly) trajectory of slope $-135°$ until it encounters row $n-1$ (the bottom row). The analysis in Sect. 1.3.1.1 ensures that this 2-stage trajectory brings $\mathcal{F}$ to the correct tile along $\mathcal{M}_n$'s bottom row for depositing object σ. After depositing σ, $\mathcal{F}$ retraces its steps back to row 0 in preparation for transporting the object that had been to the right of object σ.

Since pattern-reversal is a rather simple operation, yet not a trivial one, this is a good place to give an explicit instantiation of the generic FSM program scheme of Fig. 1.4. Figure 1.10 presents an FSM program that implements the single-FSM version of the algorithm implicit in Fig. 1.8(right); note that the notation used is a compressed version of that used in Fig. 1.4.

B. Rotational rearrangements. The *pattern-rotation* problem has $\mathcal{F}$ transport the objects from row 0 of $\mathcal{M}_n$ to row $n-1$, where it deposits them in a cyclically rotated order. The amount of rotation is specified by the index-position k along row $n-1$ where the object from tile $\langle 0, 0\rangle$ is to be deposited; cf. Fig. 1.11(left, center) for the case $k = 3$ and $n = 8$. Our focus here is on the rather challenging variant of the pattern-rotation problem in which the amount of rotation is indicated by a

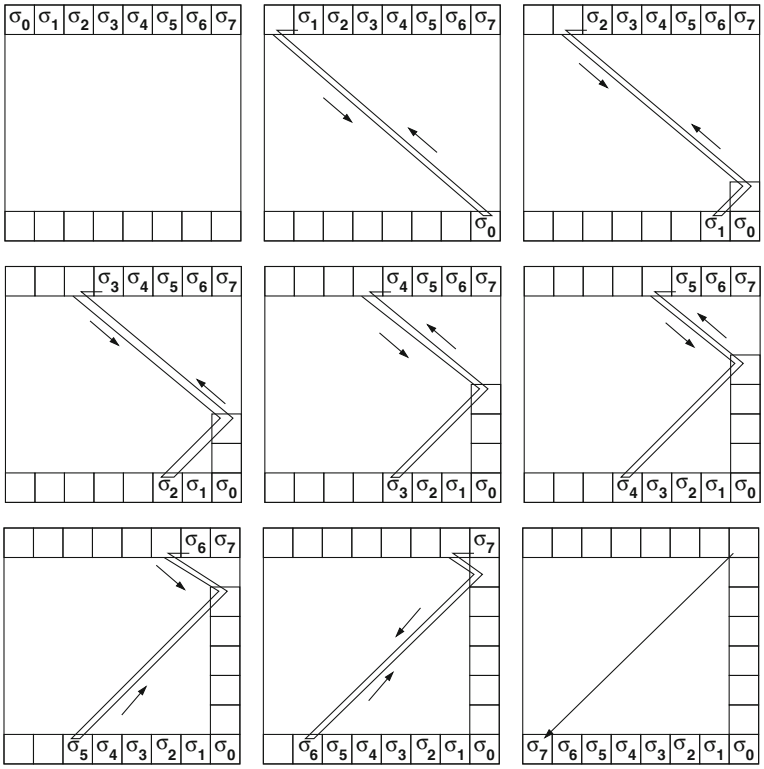


Fig. 1.9 Illustrating reversal rearrangement in action. All trajectory slopes are multiples of 45°

pre-specified rational φ in the range $0 \leq \varphi \leq 1$; the problem specified via φ mandates that the pattern be cyclically rotated $\lfloor \varphi(n-1) \rfloor$ positions as it is transported from $\mathcal{M}_n$'s top edge to its bottom edge. Within this context, Fig. 1.11(left, center) depicts a rational φ for which $\lfloor 7\varphi \rfloor = 3$. The general algorithmic strategy that enables an FSM to solve the pattern-rotation problem has the same overall structure as the strategy that works to solve the pattern-reversal problem, but in contrast to the latter problem, the pattern-rotation problem has $\mathcal{F}$ choose between two trajectory-patterns depending on which edge of $\mathcal{M}_n$ it encounters first during its trajectory from row 0: $\mathcal{F}$ follows the trajectory-pattern of Fig. 1.11(right a) when $\mathcal{M}_n$'s bottom edge is the first edge it encounters after leaving row 0; $\mathcal{F}$ follows the trajectory-pattern of Fig. 1.11(right b) when it encounters $\mathcal{M}_n$'s right edge before it encounters $\mathcal{M}_n$'s bottom edge. Actually, the two trajectory-patterns within Fig. 1.11(right) are closely related: the pattern in Fig. 1.11(right b) would be identical to the pattern in Fig. 1.11(right a) if one could extend $\mathcal{M}_n$ rightward (by adding columns). Because we cannot actually add any columns to $\mathcal{M}_n$, we instead simulate the effect of doing so by overlaying the added "suffix" of $\mathcal{M}_n$ on top of the initial columns of the $n \times n$ version. Figure 1.12 illustrates the stages as $\mathcal{F}$ implements a 3-position rotation-rearrangement of an 8-object pattern.

An FSM $\mathcal{F}$ for the Pattern-Reversal Problem

Current State	Tile Type	Action	Move Direction	Next State
→ SEEK	empty, top-edge	(none)	east (→)	SEEK
〃	object, top-edge	pick up object	south (↓)	DELIVER-SE
〃	empty, right-edge	(none)	no-move	HALT
DELIVER-SE	interior	(none)	southeast (↘)	DELIVER-SE
〃	right-edge	(none)	southwest (↙)	DELIVER-SW
DELIVER-SW	interior	(none)	southwest (↙)	DELIVER-SW
〃	bottom-edge	deposit object	northeast (↗)	RETURN-NE
RETURN-NE	interior	(none)	northeast (↗)	RETURN-NE
〃	right-edge	(none)	northwest (↖)	RETURN-NW
RETURN-NW	interior	(none)	northwest (↖)	RETURN
〃	top-edge	(none)	east (→)	SEEK

Fig. 1.10 A program for the pattern-reversal FSM $\mathcal{F}$ as it: *seeks* the next object along $\mathcal{M}_n$'s *top edge*; *picks up* the first found object (halting in the inescapable "halt state" HALT if there is none); *conveys* the object to $\mathcal{M}_n$'s *bottom edge* (via a SE-then-SW path), where it *deposits* the object; *returns* to $\mathcal{M}_n$'s *top edge* (via a NE-then-NW path) to continue the process. The *start state* SEEK is indicated by an *arrow*. Unspecified conditions—such as encountering an interior tile in state SEEK—all send $\mathcal{F}$ to state HALT

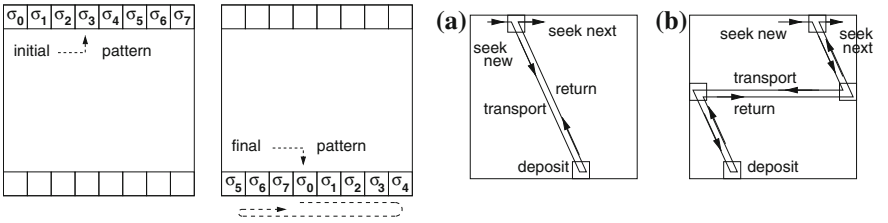


Fig. 1.11 (*left*, *center*) The initial and final configurations under the rotation rearrangement: the example illustrated is a 3-position rotation within $\mathcal{M}_8$. (*right*) A pair of generic trajectories that allow $\mathcal{F}$ to achieve fixed rotations: **a** the trajectory that $\mathcal{F}$ follows when it encounters $\mathcal{M}_n$'s bottom edge before its right edge; **b** the trajectory that $\mathcal{F}$ follows when it encounters $\mathcal{M}_n$'s *right edge* before its *bottom edge*; all diagonal paths have the same slope

1.3.2 Algorithms Based on "Hugging" $\mathcal{M}_n$'s "Walls"

The rearrangement problems discussed in Sect. 1.3.1 are, in a sense, convenient to solve because some prespecified "codeword" such as "reversal" or prespecified parameter such as the rotation-specifying rational φ identifies *ab initio* trajectory-patterns that can be used to solve any problem instance. We now discuss an algorithmic rearrangement strategy that often can be used to solve less conveniently identified rearrangement problems, specifically ones that require complex run-time adaptation of trajectory-patterns. (The word "complex" here is intended to distinguish the required adaptations from the simple binary-switch adaptation we used with the rotation-rearrangement problem in Sect. 1.3.1.3.B).

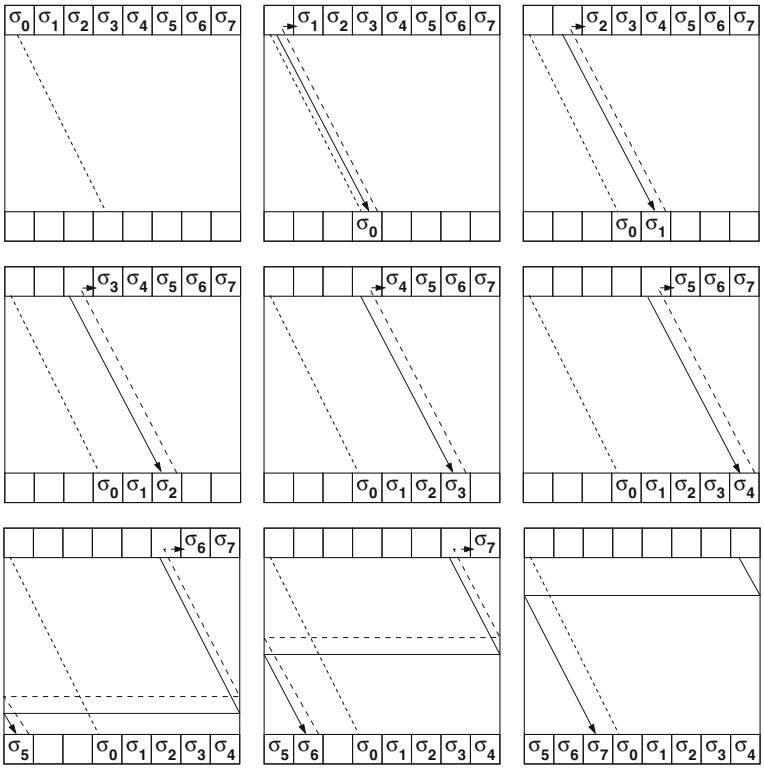


Fig. 1.12 Illustrating pattern rotation in action. The *light dashed* "guide line" is included to enhance legibility; it does not really exist

The strategy of circumnavigating $\mathcal{M}_n$ while "hugging its walls" yields a flexible tool for solving a broad range of "runtime-determined" pattern-rearrangement problems. The circumnavigatory trajectory-pattern is illustrated in its generic form in Fig. 1.13. FSM $\mathcal{F}$ makes a series of circumnavigations of $\mathcal{M}_n$, transporting one object from row 0 to row $n-1$ during each circuit. The power of this algorithmic strategy is manifest in:

- its broad applicability; in fact, it can be used to solve both of the rearrangement problems discussed in Sect. 1.3.1.3;
- the ease of having a team of FSMs achieve roughly linear parallel speedup by pipelining any circumnavigation-based algorithm.

The weakness of the strategy arises from its inflexibility regarding possible available "shortcuts": each circumnavigation takes the same amount of time—roughly $4n$ steps. In contrast, one can sometimes employ a problem-specific strategy to speed up the rearrangement by a factor of 2. The reversal-FSM of Sect. 1.3.1.3. A achieves

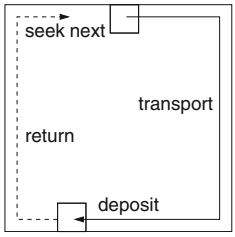


Fig. 1.13 A generic circumnavigatory trajectory that enables $\mathcal{F}$ to accomplish a variety of rearrangement operations. $\mathcal{F}$ traverses the solid path as it transports the next object from its initial *tile along row* 0 to its destination *tile along row* $n-1$. $\mathcal{F}$ traverses the *dashed path* as it returns to $\mathcal{M}_n$'s *top row* to seek the next object

such a speedup, and the rotation-FSM sometimes does—depending on the rational φ and the FSM's single-step move repertoire (cf. Fig. 1.2).

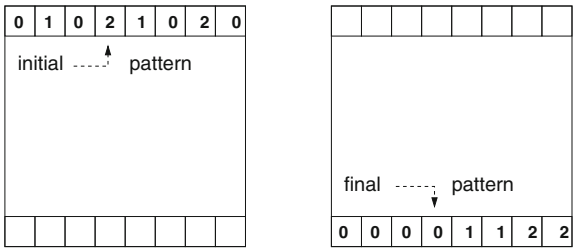
We flesh out the preceding abstract discussion by focusing on two significant "runtime-determined" pattern-rearrangement problems. The *sorting-rearrangement* problem of Sect. 1.3.2.1 calls for an FSM $\mathcal{F}$ that transports the objects that reside in row 0 of $\mathcal{M}_n$ to row $n-1$, where $\mathcal{F}$ deposits the objects in sorted order according to their types. (We assume, of course, that object-types come from an ordered set.) Clearly, $\mathcal{F}$ can determine the ultimate placement of an object σ along row $n-1$ only when it discovers how many objects along row 0 have types that are smaller than σ's (under the ordering of types). We show in Sect. 1.3.2.1 how the circumnavigation strategy yields an efficient solution to the sorting-rearrangement problem. We close this section in Sect. 1.3.2.2 by discussing a variant of the circumnavigation strategy that solves a variant of the pattern-verification problem of Sect. 1.3.1.2.

1.3.2.1 The *Sorting* Rearrangement

Figure 1.14 illustrates the initial and final patterns of a sorting problem on $\mathcal{M}_8$, wherein objects have types 0, 1, and 2, with the natural order $0 < 1 < 2$.

Figure 1.15 illustrates the stages as $\mathcal{F}$ employs the circumnavigation strategy to sort an 8-object pattern. The illustrated process is rather straightforward. $\mathcal{F}$ makes

Fig. 1.14 The initial and final configurations under the sorting operation



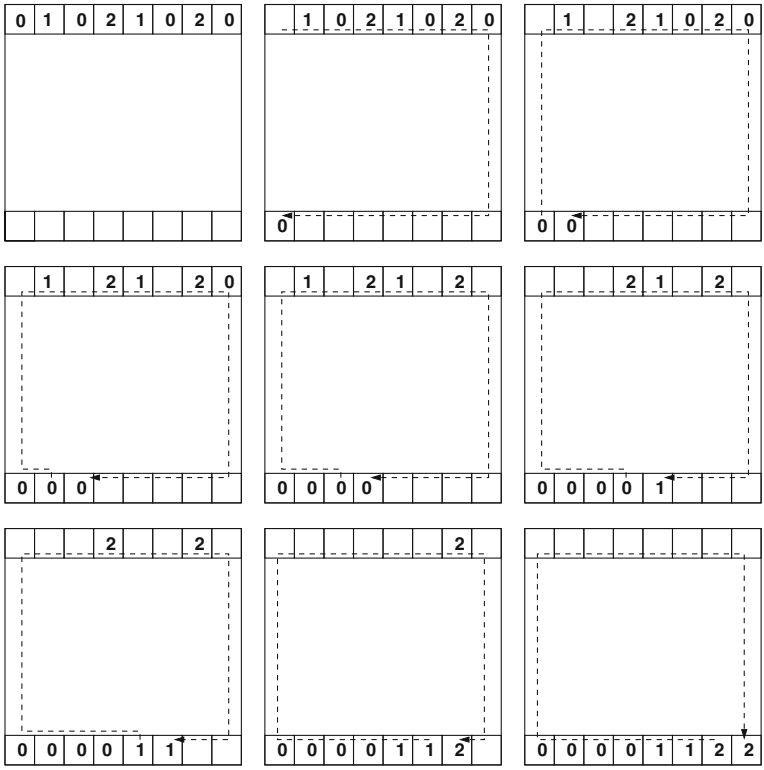


Fig. 1.15 Illustrating sorting in action

multiple passes along row 0. During the first batch of passes, each circumnavigation transports one object having the smallest order-type, call it o, to the block of leftmost tiles along row $n-1$ that are dedicated to objects of this type. $\mathcal{F}$ identifies this block by proceeding along row $n-1$ until it encounters either an object of type o or $\mathcal{M}_n$'s left edge (which signals that no objects of type o have yet been deposited along row $n-1$). Having transported all objects of type o, $\mathcal{F}$ embarks on a second batch of passes, during which it transports all objects having the second smallest type to the leftmost tiles along row $n-1$ that are to the right of the tiles that hold the objects of type o. This batched process continues, with a new batch of passes handling each successively larger object-type. One final pass that verifies the absence of remaining objects along row 0 terminates the process. Note that, with the exception of this single empty-handed circumnavigation of $\mathcal{M}_n$'s periphery, $\mathcal{F}$ performs one complete circuit per object. The entire process thus takes roughly $4n^2$ steps.

We believe, but have yet to prove that the preceding sorting-rearrangement algorithm cannot be sped up by any problem-specific strategy, because of the earlier-mentioned fact that the ultimate destination along row $n-1$ of any object σ

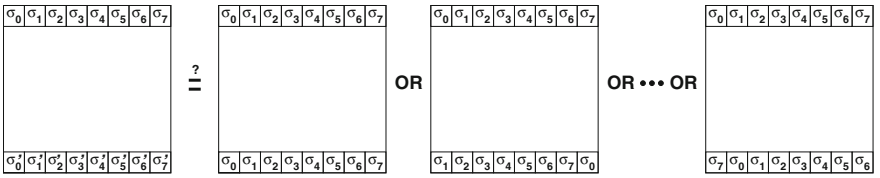


Fig. 1.16 Illustrating the rotation-recognition problem. An FSM must *scalably* determine whether, within the *leftmost subfigure*, the pattern of object-types along the *bottom row* of $\mathcal{M}_n$ is a rotation of the pattern along the *top row*. In other words, the object-laden instance of $\mathcal{M}_n$ in the *leftmost subfigure* must be one of the instances to the *right* of the equal sign

from row 0 depends on the number of objects that began along row 0 that have types smaller than σ's.

1.3.2.2 Revisiting the Pattern-Identifying/Checking Problem

Each problem of the genre discussed in Sect. 1.3.1.3 demands a single FSM that can transport the objects from $\mathcal{M}_n$'s top row to $\mathcal{M}_n$'s bottom row, depositing them there according to some specified single pattern. We turn now to a kindred genre of problem that begins with objects lining both $\mathcal{M}_n$'s top row and its bottom row. The challenge now is to design an FSM (or a team thereof) that can scalably verify whether the pattern of object-types along row $n-1$ belongs to a prespecified family of rearrangements of the pattern of object-types along row 0. Of course, if the target family of rearrangements comprises just a single genre of rearrangement—say, a pattern reversal or a pattern rotation—then solving the just-described verification problem is never more difficult than solving the associated rearrangement problem. To wit, design a rearranging FSM that then checks the results of its rearrangement against the input pattern along row $n-1$. However, if the target family comprehends a large number of possible rearrangements, then solving the verification problem can require rather different algorithmic tools than does each individual target rearrangement. This section introduces the verification problem via just one highly-populated family of target rearrangements, namely, the family of all possible pattern rotations. In detail, the problem of interest has the following form. The mesh $\mathcal{M}_n$ has objects arranged along both its top and bottom rows. One must design an FSM that can, within arbitrarily large meshes, answer the following question: *Is the pattern of object-types along $\mathcal{M}_n$'s bottom row a rotation of the pattern along its top row?* See Fig. 1.16.

It is not hard to design an FSM $\mathcal{F}$ that solves this rotation-recognition problem, especially under the assumption that all of $\mathcal{M}_n$'s tiles other than those in the top and bottom rows are empty. We make this assumption here; clerical modifications can get around it. The algorithm that our FSM $\mathcal{F}$ embodies is not elegant: it directly tests all possibilities seriatim. The only subtlety in the procedure is how $\mathcal{F}$ keeps track of the search space of all rotations. In Fig. 1.17, $\mathcal{F}$ accomplishes its bookkeeping by selectively moving objects from their home rows (which are row 0 or row $n-1$) to

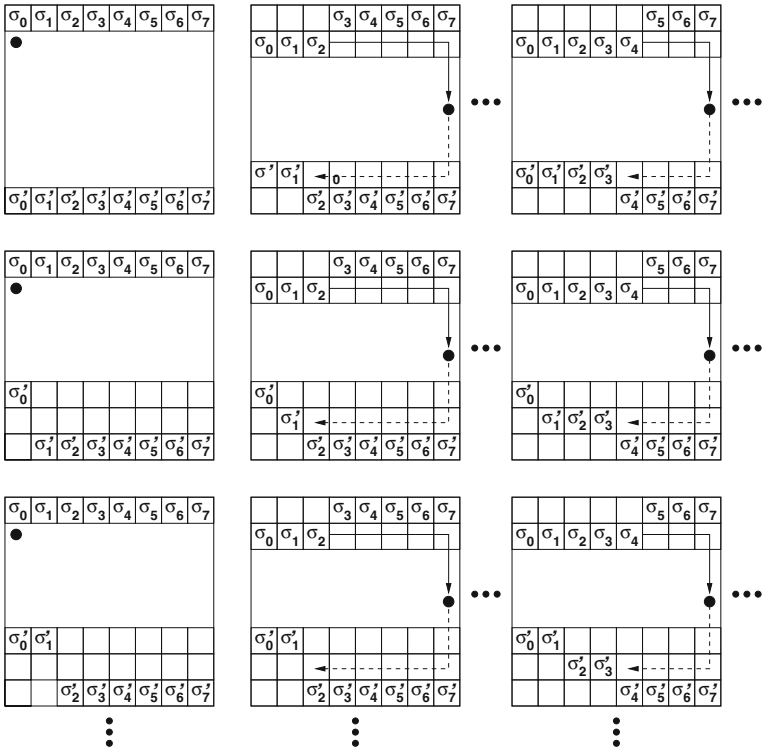


Fig. 1.17 Illustrating an FSM $\mathcal{F}$ that *scalably* solves the rotation-recognition problem. $\mathcal{F}$ must determine whether, within the mesh in the *upper-left* corner of the figure, the pattern of object-types along the *bottom row* of $\mathcal{M}_n$ is a rotation of the pattern along the *top row*. $\mathcal{F}$ achieves this by shuffling objects between the *top two rows* of $\mathcal{M}_n$ and within the *bottom three rows*; objects never leave their columns. The *third row* from the *bottom of* $\mathcal{M}_n$ (*row* $n-3$) is used to keep track of the starting places of potential rotations that have already been tested. $\mathcal{F}$ uses $\mathcal{M}_n$'s *rows* 1 and $n-2$ to keep track of its successive object-matching circumnavigations: during each circumnavigation, $\mathcal{F}$ determines whether the rightmost object of row 1 matches the leftmost object of *row* $n-1$. $\mathcal{F}$ appropriately shifts objects from *rows* 0 and $n-1$ after each match-test, to prepare for the next circumnavigation, and it appropriately shifts one object from *row* $n-2$ to prepare to test for the next potential rotation. Object-positions are reset appropriately to prepare for each new circumnavigation

neighboring rows. (For "neatness," we have $\mathcal{F}$ restore objects to their home tiles at the conclusion of the algorithm.) The algorithm proceeds in n stages, each having n substages. Recall, as in Fig. 1.16, that the sequence of object-types along row 0 is $\sigma_0\sigma_1\ldots\sigma_{n-1}$, while the sequence along row $n-1$ is $\sigma_0'\sigma_1'\ldots\sigma_{n-1}'$.

- The ith *primary stage* of the algorithm tests tile-position i along row $n-1$ to determine whether the pattern of object-types in row $n-1$ is a rotation of the pattern in row 0, with tile $\langle n-1, i\rangle$ as the *anchor* of the rotation, meaning that

$$\sigma'_0 \ldots \sigma'_{i-1}\sigma'_i \ldots \sigma'_{n-1} \;=\; \sigma_{n-i} \ldots \sigma_{n-1}\sigma_0\sigma_1 \ldots \sigma_{n-i-1}.$$

  - If this test succeeds, then $\mathcal{F}$ reports success—the pattern along row $n-1$ *is* a rotation of the pattern along row 0.
  - If the test fails, then $\mathcal{F}$ moves σ'_i up to row $n-3$, i.e., to tile $\langle n-3, i\rangle$.
    If $i = n-1$, then $\mathcal{F}$ halts and reports failure—the pattern along row $n-1$ *is not* a rotation of the pattern along row 0.
    If $i \neq n-1$, then $\mathcal{F}$ embarks upon primary stage $i+1$.

- The jth *secondary stage* of the algorithm implements the n circumnavigations that are needed to determine whether the current potential anchor tile $\langle n-1, j\rangle$ is, indeed, the *anchor* of the current rotation. For each $i \in [0, n-1]$, $\mathcal{F}$ initiates a sequence of n circumnavigations.
  The first circumnavigation begins at tile $v_0 = \langle 0, 0\rangle$. In the course of this trajectory, $\mathcal{F}$ checks whether the object at tile v_0 has the same type as the object at tile $\langle n-1, j\rangle$.
  - If the answer is "NO," then $\mathcal{F}$ aborts this primary stage and moves on to the next one.
  - If the answer is "YES," then $\mathcal{F}$ completes the circumnavigation and prepares for the next one, during which it will check whether the object at tile $\langle 0, 1\rangle$ has the same type as the object at tile $\langle n-1, j+1\rangle$.

  The important detail here is how $\mathcal{F}$ keeps track of the top and bottom tiles whose objects' types it is checking. $\mathcal{F}$ accomplishes this by moving each checked object to an adjacent row. An object from row 0 is moved to row 1; an object from row $n-c$ ($c \in \{1, 2\}$) is moved to row $n-c-1$. At the end of this secondary stage, all objects are restored to the tiles where they began this stage, in preparation for the next secondary stage.
  Importantly, the "arithmetic" that is implicitly being done to determine successive objects along the bottom rows of $\mathcal{M}_n$ is done modulo n. This is an easy finite-state operation: when $\mathcal{F}$ detects the right edge of $\mathcal{M}_n$, it begins to process tiles from the left ends of $\mathcal{M}_n$'s bottom rows rather than their middles.

One readily supplies the details that convert this sketch to a complete algorithm.

1.4 The Power of Cooperation

1.4.1 The Need for Cooperation

FSMs acting on their own suffer a fundamental limitation: No single FSM can find/identify a tile $\langle i, j\rangle$ that is "buried" within $\mathcal{M}_n$, in the sense that both i and j are far from both 0 and $n-1$; the notion "buried" is made rigorous in terms of

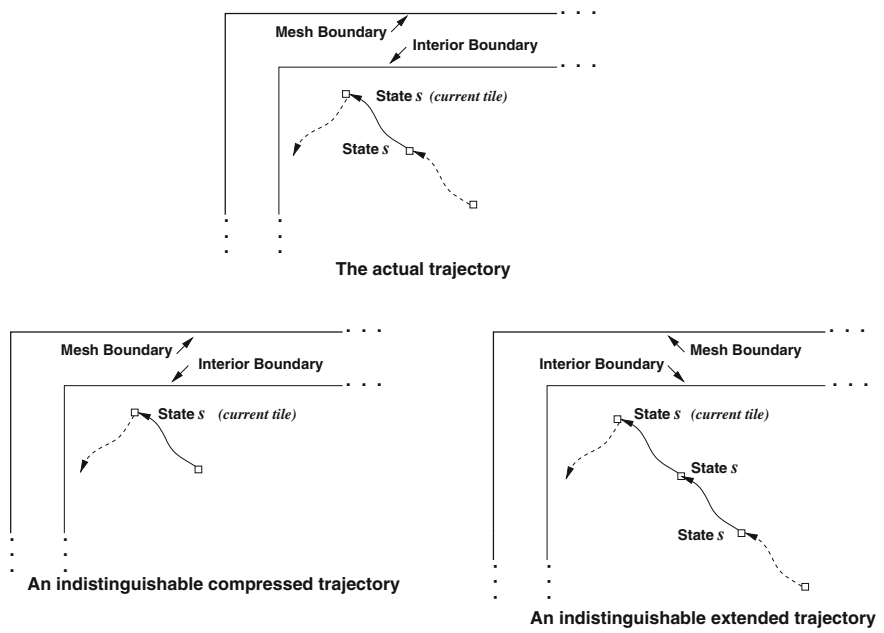


Fig. 1.18 A single FSM gets "lost" when either its start tile or its target tile is "buried" within $\mathcal{M}_n$

the number of states of the wandering FSM. The situation is illustrated schematically in Fig. 1.18, which depicts a (formalizable) intuition that can be encapsulated as follows. A q-state FSM $\mathcal{F}$ is wandering within $\mathcal{M}_n$ from an initial tile, v, toward a target tile, v'. At least one of v and v' is "buried" within $\mathcal{M}_n$: both of the "buried" tile's coordinates, i and j, satisfy $q < i, j < n - q - 1$. Along any path whose length exceeds q, $\mathcal{F}$ must enter the same state, say s, at least twice; call such a tile a *state-s tile*. If the path between any two state-s tiles lies entirely with the interior of $\mathcal{M}_n$—i.e., each tile along the path "buried" within $\mathcal{M}_n$—then $\mathcal{F}$ can never distinguish between these state-s tiles. Informally, $\mathcal{F}$ is "lost."

Sample applications of (an adequate formalization of) the preceding argument show that in sufficiently large meshes $\mathcal{M}_n$:

- $\mathcal{F}$ cannot identify which quadrant or wedge of $\mathcal{M}_n$ it resides in.
- $\mathcal{F}$ cannot find a path from, say, tile $\langle 0, 0\rangle$ to, say, tile $\langle \lfloor \frac{1}{2}n \rfloor, \lfloor \frac{1}{2}n \rfloor \rangle$. (We provide just one unfeasible source-target pair for illustration.)

Detailed versions of the sketched argument can be found in [31, 33].

The fundamental limitation of single FSMs motivates our focus in this section on teams of two or more FSMs that cooperate to solve a sampler of problems of the type we have just argued that single FSMs are unable to solve.

1.4.2 Tasks Enabled by Cooperative Behavior

Classical results from "machine-based" Computation Theory (Turing machines, Register machines, etc., as discussed in [30]) make it not surprising that teams of FSMs can cooperatively accomplish very complex tasks. What we try to suggest in this section is that the FSMs in the teams: (*a*) can often all be copies of the same FSM; (*b*) can accomplish their tasks by playing very natural roles. We focus on a few tasks of the sort that Sect. 1.4.1 shows not to be possible for single FSMs.

1.4.2.1 FSMs Exploit "Rational Tiles" Within $\mathcal{M}_n$

A. FSMs Identify "rationally Specified" Tiles

This section is devoted to the problem of having a team of identical FSMs (scalably) identify tiles of $\mathcal{M}_n$ that are specified by pairs of positive rational parameters. The instance $P(\varphi, \psi)$ of the problem, which is associated with parameters φ and ψ, requires that we design an FSM $\mathcal{F} = \mathcal{F}^{(\varphi,\psi)}$ such that teams of copies of $\mathcal{F}$ can cooperate to "identify" tile $v_{\varphi,\psi} \stackrel{def}{=} \langle \lfloor \varphi(n-1) \rfloor, \lfloor \psi(n-1) \rfloor \rangle$ of $\mathcal{M}_n$, in the sense that one designated copy of $\mathcal{F}$ halts on $v_{\varphi,\psi}$.

We describe an FSM $\mathcal{F}$ that solves problem $P(\varphi, \psi)$, in the sense that a team of three copies of $\mathcal{F}$, call them $\mathcal{F}_0$, $\mathcal{F}_1$, $\mathcal{F}_2$, can identify tile $v_{\varphi,\psi}$. The FSMs begin to solve the problem within $\mathcal{M}_n$ by moving to the first three tiles in row 0, i.e., tiles $\langle 0, 0 \rangle$, $\langle 0, 1 \rangle$, $\langle 0, 2 \rangle$. In turn, each FSM then moves to tile $\langle 0, 0 \rangle$ and follows thence a variant of the staircase trajectory of Fig. 1.3. $\mathcal{F}_0$ and $\mathcal{F}_1$ both traverse a "top to bottom" staircase from row 0 to row $n-1$ and there configure themselves as neighbors, with $\mathcal{F}_1$ at tile $\langle n-1, \lfloor \psi(n-1) \rfloor \rangle$. $\mathcal{F}_2$ traverses a "left to right" staircase from column 0 to column $n-1$, ending at tile $\langle \lfloor \varphi(n-1) \rfloor, n-1 \rangle$. The first configuration of Fig. 1.19 illustrates this initial placement of the three FSMs. In the algorithm:

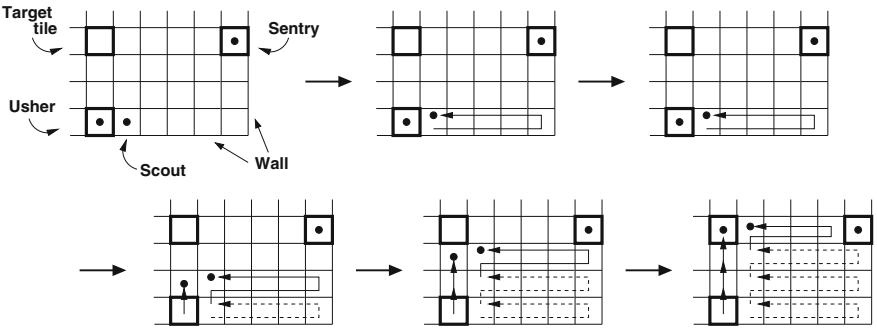


Fig. 1.19 A team of three FSMs solve the rational-point identification problem, seeking tile $\langle \lfloor \varphi(n-1) \rfloor, \lfloor \psi(n-1) \rfloor \rangle$ when $\lfloor \varphi(n-1) \rfloor$ is odd (so the implicit rectangle has an even number of *rows*). *Solid lines* depict the current trajectory; *dashed lines* depict past ones

- $\mathcal{F}_2$ acts as a *sentry* for the team, marking the northeastern corner of the rectangle whose northwestern corner is the target tile $v_{\varphi,\psi}$;
- $\mathcal{F}_1$ acts as an *usher* whose role is to guide $\mathcal{F}_0$ from its initial tile $\langle n-1, \lfloor \psi(n-1) \rfloor + 1\rangle$ to $v_{\varphi,\psi}$;
- with the help of $\mathcal{F}_1$ and $\mathcal{F}_2$, $\mathcal{F}_0$ acts as the *scout* who identifies $v_{\varphi,\psi}$.

The body of the algorithm is a sweep by $\mathcal{F}_0$ through the rectangle whose antipodal corner tiles are $\langle n-1, \lfloor \psi(n-1) \rfloor + 1\rangle$ and $\langle \lfloor \varphi(n-1) \rfloor, n-1\rangle$. The usher $\mathcal{F}_1$ moves one tile upward whenever $\mathcal{F}_0$ leaves its side, embarking on the next eastward trajectory of the sweep; $\mathcal{F}_1$ then awaits the re-arrival of $\mathcal{F}_0$ as the latter completes the next westward trajectory of the sweep. Note that $\mathcal{M}_n$'s right "wall" terminates each of $\mathcal{F}_0$'s eastward trajectories, whereupon $\mathcal{F}_0$ moves up one row and embarks on its next westward trajectory. $\mathcal{F}_2$ meanwhile awaits the arrival of $\mathcal{F}_0$; when these two FSMs meet, $\mathcal{F}_0$ knows that it is beginning its last westward trajectory. When $\mathcal{F}_0$ then meets $\mathcal{F}_1$ again (for the last time), both FSMs know that $\mathcal{F}_1$ is standing on the target tile. (One of them completes the algorithm by going back to tell $\mathcal{F}_2$ to "stand down.") The reader can easily flesh this sketch out to a complete algorithm.

The just-described version of the algorithm works when the implicit subtended rectangle has an even number of rows. We leave to the reader the easy modification needed to accommodate an odd number of rows. The only needed change involves $\mathcal{F}_0$'s final interactions with the usher and the sentry: the final segment of $\mathcal{F}_0$'s sweep will then go from the usher to the sentry rather than from the sentry to the usher. This odd-even issue arises with the next problem also, wherein a team of FSM sweeps an internal rectangular region of $\mathcal{M}_n$.

B. FSMs Sweep Rectangular and Square Regions of $\mathcal{M}_n$

The next problem we consider builds on the rational-point identification problem. The *region-sweep* problem specifies, via two pairs of positive rational parameters, $\langle \varphi_0, \psi_0\rangle$ and $\langle \varphi_1, \psi_1\rangle$ with $\varphi_0 > \varphi_0$ and $\psi_0 < \psi_1$, the antipodal corners of a rectangular region within $\mathcal{M}_n$, namely, tiles $v_0 \stackrel{def}{=} \langle \lfloor \varphi_0(n-1) \rfloor, \lfloor \psi_0(n-1) \rfloor\rangle$ (the southwest corner) and $v_1 \stackrel{def}{=} \langle \lfloor \varphi_1(n-1) \rfloor, \lfloor \psi_1(n-1) \rfloor\rangle$ (the northeast corner). The problem requires one to design an FSM $\mathcal{F}$ such that four copies of $\mathcal{F}$ can jointly sweep the rectangular region so delimited. For simplicity we describe only the situation in which the delimited rectangle has an even number of rows. Clerical changes suggested by Fig. 1.20 allow one to adapt the algorithm for an odd number of rows. (Modular arithmetic based on the four rational parameters and a sweep of one of $\mathcal{M}_n$'s rows will tell $\mathcal{F}$ whether the region has an odd or an even number of rows.)

We provide a sketch of the region-sweep procedure. Let us name the four copies of $\mathcal{F}$ and assign them roles for the sweep. $\mathcal{F}_0$ will be the *scout*; it will perform the actual sweep of the region. $\mathcal{F}_1$ and $\mathcal{F}_2$ will be the *ushers;* they will climb the vertical edges of the region at a pace that will allow them to keep $\mathcal{F}_0$ within the region.

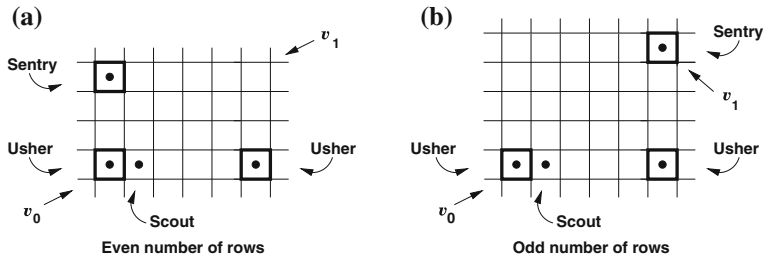


Fig. 1.20 The initial configurations for a team of 4 identical FSMs to sweep an $r \times c$ rectangular region of $\mathcal{M}_n$: the *scout* performs the sweep; two *ushers* keep the scout oriented; the *sentry* identifies the final tile. (a) The number of *rows*, r, is even; (b) r is odd

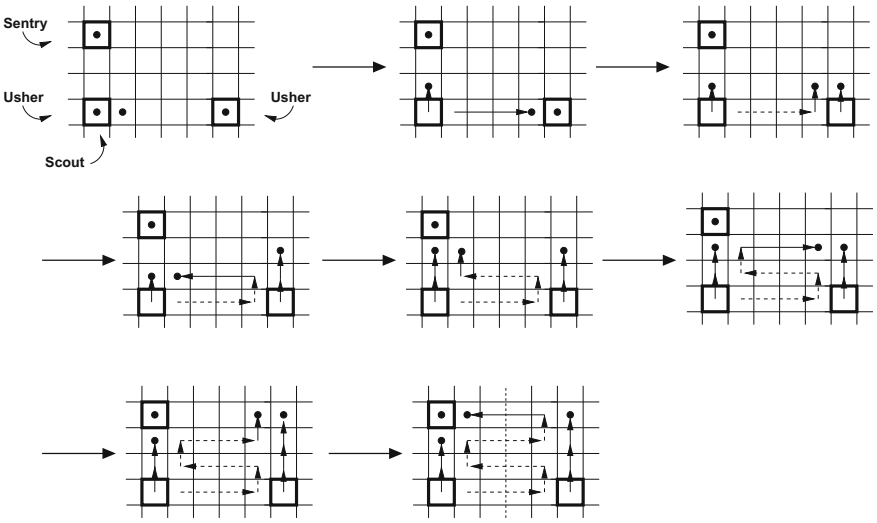


Fig. 1.21 The sweep of a 4×6 rectangular region of $\mathcal{M}_n$ by a team of 4 identical FSMs: the *scout* performs the sweep; two *ushers* keep the scout oriented; the *sentry* identifies the final tile. *Solid lines* depict the current trajectory; *dashed lines* depict past ones

$\mathcal{F}_3$ will be the *sentry*; it will find tile v_1 and stay there to tell $\mathcal{F}_0$ that it has reached the top row of the region.

The sweep of the region proceeds in the manner sketched in Fig. 1.21. Recall that we are assuming for the moment that the region has an even number of rows.

1. Using the procedure of Sect. 1.4.2.1, the team of copies of $\mathcal{F}$ position themselves within the rectangular region: $\mathcal{F}_1$ (one of the ushers) moves to tile v_0. $\mathcal{F}_0$ (the scout) moves to the right neighbor of v_0, just next to $\mathcal{F}_1$. $\mathcal{F}_2$ (the other usher) moves to the southeastern corner tile of the region, by "resolving" the coordinates of v_0 and v_1; in a similar way, $\mathcal{F}_3$ (the sentry) moves to the northwestern corner tile of the region. By synchronizing the initiations of their positioning trajectories, the FSMs can guarantee that they can begin the sweep algorithm as soon as they reach their assigned tiles.

2. $\mathcal{F}_0$ begins its first eastward sweep. $\mathcal{F}_1$ moves one tile northward as soon as $\mathcal{F}_0$ leaves its side; it will stay there until $\mathcal{F}_0$ returns to its side. $\mathcal{F}_2$ awaits $\mathcal{F}_0$'s arrival. Upon that arrival, $\mathcal{F}_0$ begins its first westward sweep. $\mathcal{F}_2$ moves one tile northward as soon as $\mathcal{F}_0$ leaves its side; it will stay there until $\mathcal{F}_0$ returns to its side.

By iterated executions of phase 2, $\mathcal{F}_0$ will alternate eastward and westward sweeps in successively higher rows of the region, while $\mathcal{F}_1$ and $\mathcal{F}_2$ climb along the region's western and eastern columns, respectively, as they keep $\mathcal{F}_0$ from leaving the region. Eventually, $\mathcal{F}_1$ will encounter $\mathcal{F}_3$. When $\mathcal{F}_0$ next returns to $\mathcal{F}_1$, the latter FSM will inform $\mathcal{F}_0$ that it is about to embark on the last eastward sweep. $\mathcal{F}_0$ uses this information to halt when it next encounters $\mathcal{F}_2$.

A glance at Fig. 1.20 should help the reader amend the preceding algorithm to accommodate rectangular regions with odd numbers of rows.

1.4.2.2 Teams of FSMs Identify Their Home Wedges

We now show how to design an FSM $\mathcal{F}$ such that two copies of $\mathcal{F}$ can, when begun on adjacent tiles of any mesh $\mathcal{M}_n$, determine their home wedges within $O(n)$ steps. (Note that the same FSM works for all meshes!)

Say that we have two copies, $\mathcal{F}_{\mathrm{L}}$ and $\mathcal{F}_{\mathrm{R}}$, of an FSM $\mathcal{F}$, which reside on adjacent tiles of a mesh $\mathcal{M}_n$. We lose no generality by positing that $\mathcal{F}_{\mathrm{L}}$ resides on some tile $\langle i, j\rangle$ of $\mathcal{M}_n$, where $j < n-1$, while $\mathcal{F}_{\mathrm{R}}$ resides on the right neighbor tile $\langle i, j+1\rangle$. Easy changes to the procedure we describe now to accommodate other adjacent configurations for $\mathcal{F}_{\mathrm{L}}$ and $\mathcal{F}_{\mathrm{R}}$.

The core of our wedge-identifying procedure is to have $\mathcal{F}_{\mathrm{L}}$ and $\mathcal{F}_{\mathrm{R}}$ each perform two roundtrip walks from its initial tile to some edge of $\mathcal{M}_n$. The outward segment of the first walk consists of a sequence of $(-1, +1)$, northeasterly, steps; the inward segment of the first walk consists of a sequence of $(+1, -1)$, southwesterly, steps. The outward segment of the second walk consists of a sequence of $(-1, -1)$, northwesterly, steps; the inward segment of the second walk consists of a sequence of $(+1, +1)$, southeasterly, steps. See Fig. 1.22.

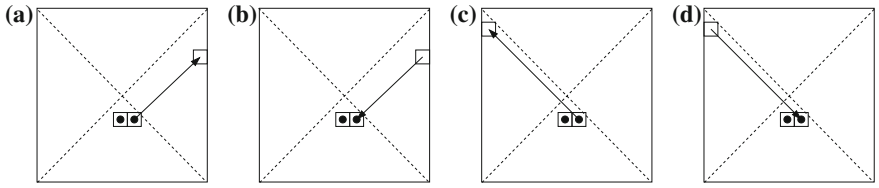


Fig. 1.22 Illustrating the roundtrip walks in the home-wedge determination algorithm. $\mathcal{F}_{\mathrm{L}}$ and $\mathcal{F}_{\mathrm{R}}$ are depicted as *bold dots* in tiles that are depicted as transparent *squares*. **a** The northeasterly outward walk; **b** the southwesterly return; **c** the northwesterly outward walk; **d** the southeasterly return. $\mathcal{F}_{\mathrm{L}}$ acts as a sentry to enable $\mathcal{F}_{\mathrm{R}}$ to "return home." The roles of the FSMs reverse when $\mathcal{F}_{\mathrm{L}}$ is performing its roundtrip walks

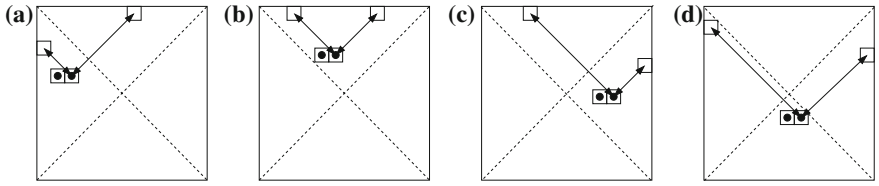


Fig. 1.23 Illustrating home-wedge determination for two adjacent FSMs. $\mathcal{F}_{\mathrm{L}}$ and $\mathcal{F}_{\mathrm{R}}$ are depicted as *bold dots* in tiles that are depicted as transparent squares. $\mathcal{F}_{\mathrm{R}}$ discovers via its pair of walks that it is a **a** "westerner"; **b** "northerner"; **c** "easterner"; **d** "southerner"

The complete algorithm has each FSM take its roundtrip walks while the other stays stationary, acting as a sentry so that the moving FSM can return to its initial tile; see Fig. 1.23. We now verify that each FSM can determine its home wedge from the termini—$\mathcal{M}_n$'s top edge or one of its side edges—of its outward walks.

We focus first on the northeasterly outward walk by an FSM $\mathcal{F}$. This walk terminates either at a tile along $\mathcal{M}_n$'s top row (Fig. 1.23a and b) or at a tile along $\mathcal{M}_n$'s rightmost column (Fig. 1.23c and d). For the purposes of ths analysis, let us consider $\mathcal{M}_n$'s northeastern tile $\langle 0, n-1\rangle$ as belonging to its top row. The SW-NE line L_1 that connects tiles $\langle n-1, 0\rangle$ and $\langle 0, n-1\rangle$ plays a major role in this case.

- In the case of a top-row terminus, $\mathcal{F}$ will have reached a tile of the form $\langle 0, k\rangle$, where $0 \le k \le n-1$. This means that $\mathcal{F}$ began the walk at a tile of the form $\langle h, k-h\rangle$, where $0 \le h \le k$, meaning that $\mathcal{F}$ began either in wedge $\mathcal{W}_N$ or wedge $\mathcal{W}_W$, i.e., "above" the line L_1.
- In the case of a right-column terminus, $\mathcal{F}$ will have reached a tile of the form $\langle k, n-1\rangle$, where $0 < k \le n-1$. This means that $\mathcal{F}$ began the walk at a tile of the form $\langle k+h, n-1-h\rangle$, where $n-1-k \le h < n-1$, meaning that $\mathcal{F}$ began either in wedge $\mathcal{W}_E$ or wedge $\mathcal{W}_S$, i.e., "below" the line L_1.

We turn next to the northwesterly outward walk by an FSM $\mathcal{F}$. This walk terminates either at a tile along $\mathcal{M}_n$'s top row (Fig. 1.23b and c) or at a tile along $\mathcal{M}_n$'s leftmost column (Fig. 1.23a and d). For the purposes of ths analysis, let us consider $\mathcal{M}_n$'s northwestern tile $\langle 0, 0\rangle$ as belonging to its top row. The NW-SE line L_2 that connects tiles $\langle 0, 0\rangle$ and $\langle n-1, n-1\rangle$ plays a major role in this case.

- In the case of a top-row terminus, $\mathcal{F}$ will have reached a tile of the form $\langle 0, k\rangle$, where $0 \le k \le n-1$. This means that $\mathcal{F}$ began the walk at a tile of the form $\langle h, k+h\rangle$, where $0 \le h \le n-1-k$, meaning that $\mathcal{F}$ began either in wedge $\mathcal{W}_N$ or wedge $\mathcal{W}_E$, i.e., "above" the line L_2.
- In the case of a left-column terminus, $\mathcal{F}$ will have reached a tile of the form $\langle k, 0\rangle$, where $0 < k \le n-1$. This means that $\mathcal{F}$ began the walk at a tile of the form $\langle k+h, h\rangle$, where $0 \le h \le n-1-k$, meaning that $\mathcal{F}$ began either in wedge $\mathcal{W}_W$ or wedge $\mathcal{W}_S$, i.e., "below" the line L_2

Table 1.1 encapsulates the results of our analysis and indicates how the outward walks enable an FSM to identify its home wedge.

Table 1.1 Inferences from each FSM's wedge-determining walks

$\mathcal{F}$'s northwesterly walk		$\mathcal{F}$'s northwesterly walk	
Walk terminated by	$\mathcal{F}$'s home wedge	Walk terminated by:	$\mathcal{F}$'s home wedge
$\mathcal{M}_n$'s corner or top edge	$\mathcal{W}_N$ or $\mathcal{W}_E$	$\mathcal{M}_n$'s corner or top edge	$\mathcal{W}_N$ or $\mathcal{W}_E$
$\mathcal{M}_n$'s left edge	$\mathcal{W}_W$ or $\mathcal{W}_S$	$\mathcal{M}_n$'s left edge	$\mathcal{W}_W$ or $\mathcal{W}_S$

Table 1.2 An FSM identifies its home wedge

$\mathcal{F}$'s home wedge	NE-walk terminus	NW-walk terminus
$\mathcal{W}_N$	corner or top edge	corner or top edge
$\mathcal{W}_E$	right edge	corner or top edge
$\mathcal{W}_S$	right edge	left edge
$\mathcal{W}_W$	corner or top edge	left edge

Table 1.2 "inverts" the information from Table 1.1 by indicating the outcomes that an FSM can expect when it begins in each of $\mathcal{M}_n$'s wedges.

1.5 Conclusion

1.5.1 Retrospective

This chapter attempts to extract and distill the lessons from our three-paper (thus far) study of the capabilities and limitations of (teams of) finite-state robots (FSMs) that operate within geographically constrained environments, modeled as square meshes. All three papers develop algorithms that are *scalable*, in the sense that they work in arbitrarily large meshes. The first two papers in the series, [31] and [33], study problems that involve only themes involving path-planning and exploration. The third paper, [34], has FSMs rearrange objects within their mesh, via algorithms that are *efficient* and *fully parallelizable* via pipelining. The specific problems studied in these sources were chosen to exercise different capabilities of FSMs, while suggesting the availability of *systematic* solutions to problems that are reminiscent of (components of) problems that one might encounter in real robotic systems.[2] We have provided here instances of all of the genres of problems studied within these papers.

In addition to distilling our earlier work, we have here begun to explore a genre of problem that is only touched on in our earlier work. We have begun to study *recognition*—or, *decision*—problems that promise to shed light on the capabilities and limitations of FSMs. Each such problem requires (teams of) FSMs to determine whether the pattern of objects residing within a mesh satisfies specific criteria.

[2] See, e.g., [16] for a view of real robotic rearrangement problems.

Although recognition problems do not generally find immediate applicability in the domain of robotics, such problems have been found, over many decades, to be an excellent vehicle for exposing the inherent nature of computational devices/systems. Indeed, both Computation Theory and Complexity Theory found their origins within the realm of recognition problems (see, e.g., [30]).

1.5.2 Prospective

Our work has just scratched the surface of an exciting and potentially applicable topic of study. We present just a few directions for future work that seem both interesting and promising.

1.5.2.1 The Inherent Limitations of FSMs

With the exception of a couple of results that depend on the fact that FSMs "lose their bearings" within the interiors of large meshes (see [31, 33] and Sect. 1.4.1), our studies have focused entirely on what (teams of) FSMs *can* accomplish, not on what they *cannot*. It would be valuable to expose the limitations of FSMs within the contexts of a broad range of activities. Such activities would involve more complicated problems involving exploration in large empty spaces than we have considered thus far, including the limitations of *teams* of FSMs, not just individual ones. The activities must also include problems that require FSMs to move and/or manipulate objects.

1.5.2.2 More Complicated Settings

All of the problems we have studied have a regular structure, involving teams of identical FSMs cooperating to perform identical (or very similar) tasks. Such regularity distances these problems from the highly irregular problems that real robotic systems must cope with; cf. [16]. Ultimately, one wants to find parallelizable algorithms for possibly heterogeneous teams of FSMs that accomplish irregularly structured tasks. Perhaps one could even have heterogeneous teams of FSMs somehow learn which FSMs are better at various tasks—a type of specialization. Along not-dissimilar lines, it would be valuable to adapt the insights, tools, and results from sources such as [7, 12, 13, 19, 20, 29] to generate and study a repertoire of recognition problems that incorporate the spirit of robotic applications.

References

1. Adler, F., Gordon, D.: Information collection and spread by networks of patrolling ants. Am. Nat. **140**, 373–400 (1992)
2. Basu, P., Redi, J.: Movement control algorithms for realization of fault-tolerant ad hoc robot networks. IEEE Netw. **1**8(4), 36–44 (2004)
3. Bender, M., Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. In: 35th IEEE Symposium on Foundations of Computer Science, pp. 75–85 (1994)
4. Bhatt, S., Even, S., Greenberg, D., Tayar, R.: Traversing directed eulerian mazes. J. Graph Algorithms Appl. **6**, 157–173 (2002)
5. Blum, M., Sakoda, W.: On the capability of finite automata in 2 and 3 dimensional space. In: 18th IEEE Symposium on Foundations of Computer Science, pp. 147–161 (1977)
6. Böhringer, K.F.: Modeling and controlling parallel tasks in droplet-based microfluidic systems. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, pp. 329–339 (2006)
7. Borchert, B., Reinhardt, K.: Deterministically and sudoku-deterministically recognizable picture languages. In: 2nd International Conference on Language and Automata Theory and Applications (LATA'07) (2007)
8. Budach, L.: On the solution of the labyrinth problem for finite automata. Elektronische Informationsverarbeitung und Kybernetik (EIK) **11**(10–12), 661–672 (1975)
9. Chowdhury, D., Guttal, V., Nishinari, K., Schadschneider, A.: A cellular-automata model of flow in FSM trails: non-monotonic variation of speed with density. J. Phys. A Math. Gen. **35**, L573–L577 (2002)
10. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Label-guided graph exploration by a finite automaton. ACM Trans. Algorithms **4**, 1–18 (2008)
11. Geer, D.: Small robots team up to tackle large tasks. IEEE Distrib. Syst. Online **6**(12), 2 (2005). doi:10.1109/MDSO.2005.66
12. Giammarresi, D., Restivo, A.: Recognizable picture languages. Int'l. J. Pattern Recogn. Artif. Intell. **6**(2–3), 241–256 (1992)
13. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages III, pp. 215–267. Springer, Heidelberg (1996)
14. Goles, E., Martinez, S. (eds.): Cellular Automata and Complex Systems. Kluwer, Amsterdam (1999)
15. Gruska, J., La Torre, S., Parente, M.: Optimal time and communication solutions of firing squad synchronization problems on square arrays, toruses and rings. In: Calude, C.S., Calude, E., Dinneen, M.J. (eds.) Developments in Language Theory, pp. 200–211. Lecture Notes in Computer Science, vol. 3340, Springer, Heidelberg (2004)
16. http://www.kivasystems.com/
17. Kobayashi, K.: The firing squad synchronization problem for two-dimensional arrays. Inf. Control **34**, 177–197 (1977)
18. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Ann. Math. Artif. Intell. **31**, 41–76 (2001)
19. Latteux, M., Simplot, D.: Context-sensitive string languages and recognizable picture languages. Inf. Comput. **138**, 160–169 (1997)
20. Latteux, M., Simplot, D.: Recognizable picture languages and domino tiling. Theor. Comput. Sci. **178**(1–2), 275-283 (1997)
21. Lothaire, M.: Combinatorics on words. In: Lyndon, G.-C., Rota, R. Lyndon, (eds.) Cambridge Mathematical Library, vol. 17. Cambridge University Press, Cambridge (1997)
22. Marchese, F.: Cellular automata in robot path planning. In: EUROBOT'96, pp. 116–125 (1996)
23. Mazoyer, J.: On optimal solutions to the firing squad synchronization problem. Theor. Comput. Sci. **168**(2), 367–404 (1996)
24. Milgram, D.L.: A region crossing problem for array-bounded automata. Inf. Control **31**(2), 147–152 (1976)

25. Moore, E.F.: Gedanken experiments on sequential machines. Automa Studies. In: Shannon, C.E., McCarthy, J. (eds.) Annals of Mathematics Studies, vol. 34, pp. 129–153. Princeton University Press, Princeton (1956)
26. Moore, E.F.: The firing squad synchronization problem. In: Moore, E.F. (ed.) Sequential Machines, Selected Papers, pp. 213–214. Addison-Wesley, Reading (1962)
27. Müller, H.: Endliche Automaten und Labyrinthe. Elektronische Informationsverarbeitung und Kybernetik (EIK) **11**(10–12), 661–672 (1975)
28. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. **3**, 114–125 (1959)
29. Reinhardt, K.: On some recognizable picture-languages. In: Brim, L. (ed.) 23rd Conference on Mathematical Foundations of Computer Science, pp. 760–770. Lecture Notes in Computer Science, vol. 1450, Springer, Heidelberg (1998)
30. Rosenberg, A.L.: The Pillars of Computation Theory: State, Encoding, Nondeterminism. Universitext Series, Springer, Heidelberg (2009)
31. Rosenberg, A.L.: The parking problem for finite-state robots. J. Graph Algorithms Appl. **16**(2), 483–506 (2012)
32. Rosenberg, A.L.: Cellular ANTomata. Adv. Complex Syst. **15**(6) (2012)
33. Rosenberg, A.L.: Region management by finite-state robots. Comput. J. **57**(1), 59–72 (2014). doi:10.1093/comjnl/bxs150
34. Rosenberg, A.L.: Finite-state robots in a warehouse: achieving linear parallel speedup while rearranging objects. In: 42nd Intenational Conference on Parallel Processing (ICPP) (2013)
35. Russell, R.: Heat trails as short-lived navigational markers for mobile robots. In: International Conference on Robotics and Automation, pp. 3534–3539 (1997)
36. Shinahr, I.: Two- and three-dimensional firing-squad synchronization problems. Inf. Control **24**, 163–180 (1974)
37. Spezzano, G., Talia, D.: The CARPET programming environment for solving scientific problems on parallel computers. Parallel Distrib. Comput. Pract. **1**, 49–61 (1998)
38. von Neumann, J.: In: Burks, A.W. (ed.) The Theory of Self-reproducing Automata. University of Illinois Press, Urbana-Champaign (1966)
39. Wolfram, S. (ed.): Theory and Application of Cellular Automata. Addison-Wesley, Reading (1986)

Chapter 2
Lattice Automata for Control of Self-Reconfigurable Robots

Kasper Stoy

Abstract Self-reconfigurable robots are built from robotic modules typically organised in a lattice. The robotic modules themselves are complete, although simple, robots and have onboard batteries, actuators, sensors, processing power, and communication capabilities. The modules can automatically connect to and disconnect from neighbour modules and move around in the lattice of modules. The self-reconfigurable robot as a whole can, through this automatic rearrangement of modules, change its own shape to adapt to the environment or as a response to new tasks. Potential advantages of self-reconfigurable robots are extreme versatility and robustness. The organisation of self-reconfigurable robots in a lattice structure and the emphasis on local communication between modules mean that lattice automata are a useful basis for control of self-reconfigurable robots. However, there are significant differences which arise mainly from the physical nature of self-reconfigurable robots as opposed to the virtual nature of lattice automata. The problems resulting from these differences are mutual exclusion, handling motion constraints of modules, and unrealistic assumption about global, spatial orientation. Despite these problems the self-reconfigurable robot community has successfully applied lattice automata to simple control problems. However, for more complex problems hybrid solutions based on lattice automata and distributed algorithms are used. Hence, lattice automata have shown to have potential for the control of self-reconfigurable robots, but still a unifying implementation based on lattice automata solving a complex control problem running on physical self-reconfigurable robot is yet to be demonstrated.

2.1 Self-Reconfigurable Robots

The self-reconfigurable robot community grew out of the distributed autonomous robot systems community. The idea was that if multiple robots could automatically form physical bonds between each other the combined robot collective could adapt its shape and functionality in response to the environment and tasks. The basic scenario

K. Stoy (✉)
IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
e-mail: ksty@itu.dk

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_2

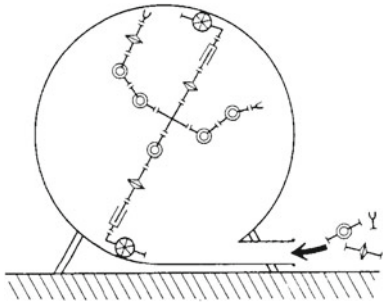


Fig. 2.1 This figure shows the original scenario that motivated the need for self-reconfigurable robots (Courtesy of Fukuda, © 1988 IEEE)

that motivated self-reconfigurable robotic research given by Fukuda et al. [8], shown in Fig. 2.1, was that individual robots could move into a storage tank through a narrow passage and once inside they could assemble for the purpose of cleaning the storage tank.

This vision of self-reconfigurable robots was and is still today attractive. However, the scientific challenges involved in realising this vision are significant. One aspect is the mechatronic realisation of self-reconfigurable robots and another central to the topic of this chapter is the question of their control.

In many self-reconfigurable robots modules are organised in a lattice structure like atoms in a crystal. These are called lattice-type self-reconfigurable robots. In these robots modules can move between lattice positions and thereby change the overall shape of the robot. The lattice organisation simplifies control of self-reconfiguration, because assumptions can be made about the precise position of neighbour modules and hence connection between modules can be performed open-looped.

Given the lattice organisation of self-reconfigurable robots, lattice automata are a natural basis for their control. However, another equally attractive feature of lattice automata is that each individual automaton acts independently and autonomously. This is crucial for self-reconfigurable robots because decoupling the controllers of individual modules will make the robot more robust to failures. A single failed module will not cause the whole system to fail which, for instance, is the case for centralised control strategies. Another desirable characteristic of lattice automata is the locality of their rules. Typically, the rules only consider the position and state of neighbour cells. These rules have a natural mapping to the sensors and the communication system that modules have which only provide functionality for inter-module communication and detection.

Given the match between the features of lattice automata and requirements of self-reconfigurable robots, researchers enthusiastically applied lattice automata to self-reconfigurable robots. Early work demonstrated how simple local rules with some noise added could make a desired configuration emerge through self-reconfiguration [11]. Another focus was the use of lattice automata to allow a self-reconfigurable robots to perform locomotion by moving modules from the back of the robot to the front [3, 13].

While this work demonstrated the potential of a lattice automata-based approach control, there is still a risk that using this hand-coded rulesets the self-reconfigurable robot could reach dead states where no rules applied. In order to be sure no dead states existed proofs were developed for specific rulesets [5, 20]. Another practical problem was the development of the rule-sets by hand, which for physical robots became quite large (e.g. 927 rules in [13]). It then became crucial to develop methods that could automatically generate rule-sets given a desired behaviour. A possibility that was explored was the use of reinforcement learning [19] and evolutionary algorithms [12]. However, both for the human rule-set designer and the automatic algorithm it became difficult to device rule-sets bottom-up given a complicated task due to combinatorial explosion of the configuration space.

A possible answer is to only control critical parts of the self-reconfiguration and allow a looser, distributed control algorithm to control the rest of the self-reconfiguration process. This simplification makes it possible to make a global-to-local compiler that based on a three-dimensional shape could generate a set of rules that would realize this shape [16]. A useful extension is to have strict rule-sets in critical areas of the robot (e.g. where there was a risk that modules may be disconnected from the structure) and let the modules move randomly in other areas [15].

Most of this work is concerned with controlling the self-reconfigurable robot itself without considering the potential of having the robot adapting to its environment. A notable exception is Bojinov et al. [1] who used rules with conditions based on the neighbour being an obstacle to creating grasping hands and other interesting functional structures. However, this line of work has not been picked up again. Lattice automata also lost traction in the self-reconfigurable robotics community because it had not been possible to create mechatronic modules that reliably could produce the motions used in the lattice automata model (e.g. rotate around a neighbour module, slide along a surface of modules, etc.). Hence, there is to this day a worry that a lattice automata based algorithm would never find practical use on a physical system. However, this may change as new mechatronic implementations are emerging that do in fact implement these motion primitives [14]. Hence, this is an exciting time for self-reconfigurable robots and lattice automata because they may finally come together to form the basis for controllable and useful self-reconfigurable robots.

2.1.1 Origin, Features, and Applications

The concept of self-reconfigurable robots was from the beginning inspired by multicellular organisms [7, 9]. The idea being that from a limited number of cell types a huge number of organisms are and can be created. For instance, an organism as complex as the human consist of about 100 trillion cells, but there are only two hundred different cell types. Hence, from an engineering perspective you could design and implement a few robotic cell types and then on the fly assemble them into a specific robot depending on your need. This concept is in the self-reconfigurable robotic community referred to as versatility. A shortcoming of mechatronic modules

is that unlike natural cells they cannot grow and divide hence another mechanism is needed to simulate growth. The mechanism used is self-reconfiguration. Instead of modules dividing and growing the robot can change its shape by rearranging the way modules are connected. In other words, modules can wander on the surface of other modules. While this is less common in nature it does happen and is known as morphallaxis. The typical example is the small fresh water animal Hydra which if cut in two can reorganize its tissues to become two hydras of roughly half the original size. As a side point there is also evidence that suggests that Hydras do not age. In fact, it is a truly remarkable animal.

Another feature of self-reconfigurable robots is robustness. Given the robot consists of many independently functioning modules, failure of one module is not critical to the functionality of the whole robot. Even if a module is placed in a critical region of the robot e.g. connecting two parts, it may be possible to replace it through self-reconfiguration. The conceptual robustness of the system of course also requires the controller to be distributed otherwise the control reduces the robustness of the entire system.

A self-reconfigurable robot is based on only a few different module types and these module types can be mass-produced. Hence, although the assembled robot is quite complex the cost of individual modules can be kept relatively low.

Together these features could provide us with robot technology that is particularly well suited for applications where the tasks are not known in advance, where the transportation cost of equipment is significant, and where robustness is important. A clear application is extra planetary exploration, but currently the most successful modular robotics company is creating robots for educational and entertainment purposes.[1]

2.1.2 Mechatronic Implementation

We have so far discussed self-reconfigurable robots at the conceptual level, but not how they are implemented in practice.

A module of a self-reconfigurable robot is a complete robot by itself. Typically, a module has sensors, actuators, processor, battery, and means of communication:

- Sensors are often limited to infrared transceivers that allow modules to detect nearby obstacles. These transceivers also often are used to communicate with neighbour modules. Otherwise, sensors are mostly internal and include encoders and accelerometers.
- Actuators include various forms of electrical motors to control the motion of a module as well as connector mechanisms.
- Processors used to be relatively small, embedded ones, but given the advance in terms of energy efficiency and computation power processors employed today

[1] http://modrobotics.com, [Online], retrieved 28/1/2013.

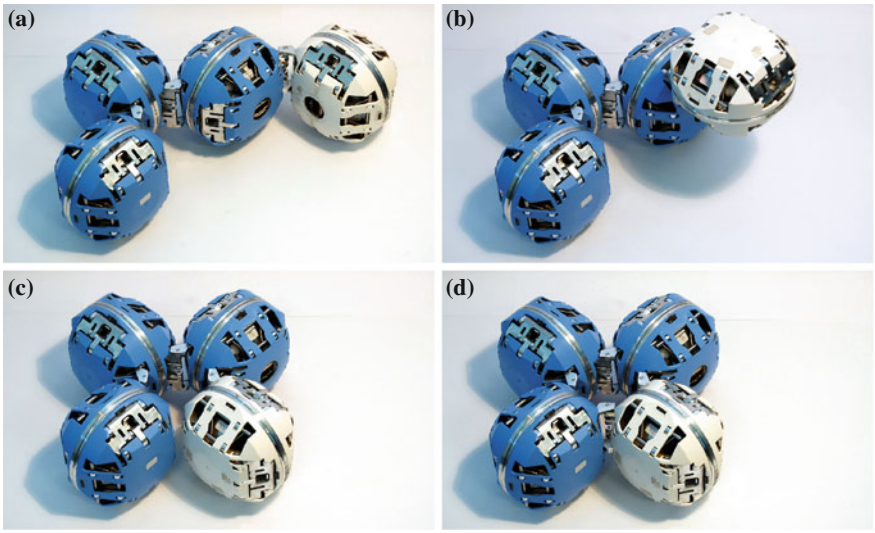


Fig. 2.2 The ATRON self-reconfigurable robot is performing one self-reconfiguration step. **a-b** First, the *top right*, *dark* module rotates the *white* module to its new position. **c-d** Once at the new position the *white* module extends connectors to attach to the new neighbour module as can be seen in the *bottom right* photo between the *dark* and *white* module

typically provide enough computation power to be able to run embedded variants of Linux.
- Batteries are typically Li-Ion allowing modules to functioning for an hour or two.
- Communication may be based on infrared communication, Bluetooth, or WiFi communication. In some cases electrical contact is made between neighbour modules allowing the modules to communicate across a shared CAN bus or similar technology.

In a typical self-reconfigurable robot, a self-reconfiguration step consists of a series of small steps as illustrated in Fig. 2.2. First a module disconnects from some of its neighbours, it then moves to a neighbour lattice position, and, finally, extends connectors to attach to neighbour modules at the new position.

The mechatronics design of self-reconfigurable robots is a major challenge given the physical constraints and the high requirements in terms of functionality. However, mechatronics is not the focus of this chapter so the interested reader can find more information on this topic in [10, 18].

2.2 Assumptions of Lattice Automata

As we have already argued lattice automata have features that match the desired features of a controller for self-reconfigurable robots. However, the match is not perfect. In fact, there are several problems one has to consider when applying lattice

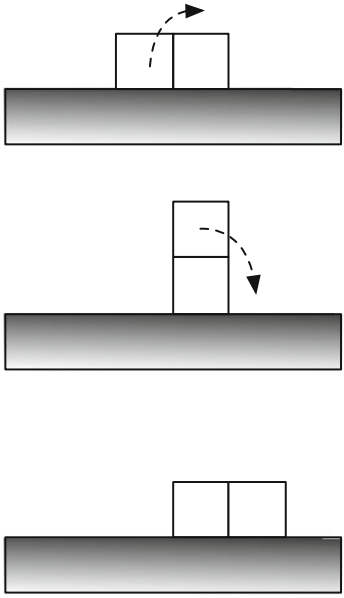


Fig. 2.3 This figure shows the two steps necessary to move a simulated self-reconfigurable robot consisting of two modules, represented by the *squares*, one step forward

automata as a basis for control of self-reconfigurable robots. We will introduce these challenges here and then continue to the actual application of lattice automata in the context of self-reconfigurable robots in the following sections.

Let us start by considering the simple motion sequence shown in Fig. 2.3. In this figure, two modules represented by squares are resting on the ground. The two-module self-reconfigurable robot can move forward by moving the rear module on top of the front module and then down in front of the module it is now resting on. This movement recreates the start configuration of the self-reconfigurable robot and the motion sequence can be repeated to generate forward locomotion. Implementing a controller based on lattice automata that realizes this concept is very simple. However, before we do this let us introduce some basic notation. There are eight directions that are relevant to these modules let us use the compass directions to identify those. For example, NE—north-east—is up and to the right of the current module. We now define the states of lattice positions, which can either be *Empty* or *Module*. Finally, we define a function *State* that maps a direction to a state. Using this notation the lattice automata rules resulting in the self-reconfiguration sequence shown in Fig. 2.3 could be as follows:

$$if\,(State(E) == Module)\;\; Move(NE) \tag{2.1}$$

$$if\,(State(S) == Module)\;\; Move(SE) \tag{2.2}$$

This may appear trivially correct, but it could in fact go wrong if there are more modules present. Even though, cell positions in a virtual world or simulator can be turned on or off easily, this is not the case here because the physical modules cannot be turned on and off. We have to simulate on and off by moving the modules. Hence, when one is turned off a neighbour cell has to be turned on. We also need to ensure that the cells a module passes through to reach the new *on* cell are free. Hence a ruleset like this is necessary:

$$if \begin{pmatrix} State(E) == Module \wedge \\ State(N) == Empty \wedge \\ State(NE) == Empty \end{pmatrix} \; Move(NE) \tag{2.3}$$

$$if \begin{pmatrix} State(S) == Module \wedge \\ State(E) == Empty \wedge \\ State(SE) == Empty \end{pmatrix} \; Move(SE) \tag{2.4}$$

This set of rules assumes that the physical module has to pass through a cell to get to the desired cell. Although this appears to be a sound rule set there are still problems. One problem is that a module does not transition instantly from one cell to another; in fact, physical modules typically take on the order of several seconds to make a transition. The implication of this is that there is significant period of time between a module detects a cell to be *Empty* and it has moved into this cell. This opens up for problems as other modules may perceive the cell as *Empty* and start a transition into it, leading to several modules moving into the same cell. From a lattice automata point of view and from the point of view of the moving module in our simple example a trivial solution could be that all cells a moving module has to pass through are marked as *Module* while the module is transitioning. However, this is impossible, because a cell can only be marked if there is a physical module in the cell able to transmit this information. A possible way to reduce this problem is to place proximity sensors in such a way that they minimize the amount of time a cell is wrongly categorised. A module leaving a cell is sensed as late as possible and a module entering a cell is sensed as early as possible. However, this of course does not solve the problem, but only reduce the probability that it will happen. One approach to solve this is to use communication to perform mutual exclusion, but in the worst case this requires time proportional to the number of modules because in a ring configuration with one hole the whole ring has to be informed. In order to completely solve this problem the lattice automata controller has to be complemented with a global communication and coordination mechanism. However, a better alternative is probably to use a roll-back strategy where modules attempt to move into cells as directed by the lattice automata, but if the module senses collision with another module moving into the same space it can roll back to its original position [6]. This might in rare cases lead to dead locks if there is a cycle, but the approach is attractive because it does not rely on global information.

Another complication is that the assumption of shared knowledge of orientations is not trivial for modular robots. Either this information has to come from embedded sensors, accelerometer and gyroscopes, or though a global coordination scheme.

Finally, several other practical issues also need to be handled: (1) one has to ensure that modules do not become disconnected from the configuration during the self-reconfiguration process because this may cause them to fall down and break, (2) one has to ensure that modules do not create hollow sub configurations that cannot be filled or where modules can be trapped.

In summary, when applying lattice automata to modular robotics one has to be concerned about the following problems.

- Motion constraints
- Mutual exclusion
- Spatial orientation
- Disconnection
- Hollow configurations

These problems arise from the physical nature of the modules and their ability to move in parallel. If not handled carefully, the problems may require global information, which again may reduce the responsiveness, scalability and robustness of the self-reconfigurable robot. While these problems are important, discussing them in detail is outside the scope of this brief introduction, but please refer to [16, 18] for details.

2.3 Lattice Automata-Based Control

Despite the complications outlined in the previous sections, many interesting examples of the use of lattice automata in the context of self-reconfigurable robots can be found in the literature.

One of the most thoroughly studied examples is cluster-flow locomotion of cubic self-reconfigurable robots, which is a generalisation of the trivial example we gave in the previous section. Butler et al. have in a series of papers extended the use of lattice automata rules for cluster-flow starting in two dimensions [3] and in later work moving to three dimensions. The algorithms were also able to handle obstacles in the environment [4]. A nice aspect of this work is also that since it has a strong theoretical basis in lattice automata it was possible to proof sufficiency and correctness for some of the rule sets. A fundamental problem, however, is that as the task and configurations become more varied the rule sets become complex. This was also a problem encountered by Østergaard et al. working with a more physical realistic simulation of the ATRON self-reconfigurable robot. In fact, for simple cluster flow locomotion 927 rules were necessary [13].

The use of hand-coded lattice automata rules becomes intractable as the complexity of the robot and the task increases. Hence, some effort has been made to automate

the design of lattice automata rules. Varshavskaya et al. [19] used a reinforcement learning based approach and Østergaard et al. [13] tried to use evolutionary algorithms. Both approaches were successful for relatively small tasks, but inherently suffer from scalability problems due to the size of the search space.

The lesson learned from this work is that lattice automata are useful if the configuration space is relatively small, but otherwise become impractical.

2.4 Hybrid Control

We have previously focused on locomotion through water-flow, however, in some cases it is not the function that is important, but the shape. Hence, transforming a self-reconfigurable robot into a specific shape becomes a task in its own right and is also important as a fundamental primitive that can be used as part of higher-level functionality. In self-reconfigurable robotics the transformation or self-reconfiguration problem is often stated as "Given an initial configuration and a goal configuration, find a sequence of module moves that will reconfigure the robot from the initial configuration to the goal configuration." [18]. Part of the effort to solve this problem has been to make global-to-local compilers. That is, compilers that take a high-level, centralised representation of a goal configuration and automatically compiles it into a lattice automata ruleset running distributedly on the modules.

Stoy et al. [16] use an approach where a three-dimensional CAD model is converted into lattice automata rules. We will describe this approach in detail below.

The first step is that a CAD model and a starting point contained inside this CAD model is given. At this point a cube is placed that represents a module. The algorithm then proceeds by adding neighbour cubes to this initial cube, but only if the position of the neighbour cube is also contained in the CAD model. Under the same conditions neighbour cubes are added to these cubes. From this point the algorithm continuous recursively until the contained volume has been filled. In effect, the surface-based representation of the CAD model is turned into a voxel-based representation. The position of each voxel corresponds to the desired position of a module in the goal configuration.

In the second step this voxel representation is turned into a set of lattice automata rules. The cubes are assigned a unique ID, in practice, an integer between 0 and N where N is the number of cubes contained in the representation. The second step is that for each pair of cubes with a shared face two lattice automata rules are generated. The direction orthogonal to the face determines the direction in which the two modules are connected. One rule creates one cube if the other cube is present and the other rule the other way around. Since they are symmetric let us just look at one rule. Let us assume we have two neighbour cubes with IDs $k1$ and $k2$ and the direction d is orthogonal to their shared face pointing from $k2$ to $k1$. If we then are in a situation where the cube with the ID $k1$ has been assigned we can then define that:

```
if (<module in direction d  equals k1 >) {
  <current module is assigned ID k2 >
}
```

We can now imagine a situation where a lattice automaton runs inside each cube with this rule set. If no IDs are assigned nothing happens. However, as soon as one module is given an ID the original assignment of IDs to cubes will be recreated.

The third step is to start a self-reconfigurable robot in a random configuration and give each module a copy of the above generated ruleset. We then trigger the self-reconfiguration process by assigning an ID to a random module. From this module IDs are assigned to neighbour modules and in turn to their neighbours and so on. Once this process has completed all modules either have an ID and belong to the goal configuration or they have no ID and are outside the goal configuration. The self-reconfiguration algorithm could now be made to work by letting modules without IDs move around randomly on the structure of module who has been assigned IDs. This would, given significant time, realize self-reconfiguration from the initial random configuration to the desired goal shape specified by the CAD model.

While acceptable in theory, the random walk of unassigned modules is not practically acceptable as the convergence to the desired configuration would take too long. In particular, when one considers that each self-reconfiguration step takes on the order of a couple of seconds. Hence, in order to speed up this process a separate algorithm is introduced that allow modules to travel to *growth points* more rapidly. A growth point in this context is a point where the neighbour module knows from the ruleset that a module should be present, but it is not. This implies that the configuration should be extended or grown in this direction. The intuition of the algorithm for attracting modules is fairly simple. A module, which is a neighbour of a growth point, sends out an integer. All modules listen for integers for a short period and propagate the smallest one they hear in this period plus one. This creates a gradient in the configuration. Now spare modules moving around in the structure can descend this gradient to locate the growth point. Once the growth point has been filled the initial module will stop transmitting and the gradient will adapt to create a gradient to a growth point further away or if no grow points are left slowly level out, because modules keep counting each other up until a maximum is reached.

Two important aspects of this approach is the division between the two control mechanisms. On one hand there are the lattice automata rules that handle the critical coordination element, e.g. ensure that the desired configuration is built, while on the other hand there is a simple algorithm for moving modules around on a global scale where the precise movement is less critical. This seems to be a critical aspect of applying lattice automata to self-reconfigurable robots because it is not tractable to encode rules for every configuration that the robot can assume. As a side point, careful configuration enumeration of the cube model has shown that for just 12 cubic modules there are more than 18 million different non-isomorphic configurations [17].

While the split into a local and a global control strategy is important, it does not have to be done as described above. Another approach explored by Rosa et al. [15] is to consider movement in the internal homogenous interior of the self-reconfigurable

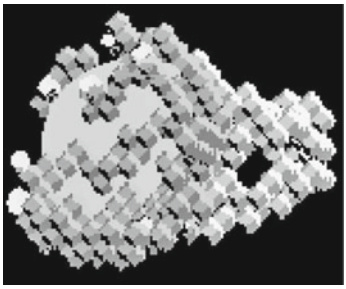


Fig. 2.4 This figure shows a self-reconfigurable robot reacting to and obstacle and grasping it. (Courtesy of Bojinov, © 2000 IEEE)

robot as part of a simple global coordination strategy while on the edges specific lattice automata rules were implemented to ensure that modules would not disconnect from the edge of the configuration which, if implemented in a physical system, would cause modules to fall down and potentially break.

Bojinov et al. [1, 2] also explore a combination of gradients for global coordination and lattice automata rules for local coordination. However, in contrast to the above work their rules are based on interaction with surrounding obstacles. For instance, if a module touches an object it can attract other modules to grow a structure around an object as shown in Fig. 2.4.

Considering this volume of work on lattice automata in the self-reconfigurable robotic community it is evident that lattice automata rarely can stand alone as a unifying control strategy for self-reconfigurable robots. However, it has been demonstrated that it is a powerful mechanism for controlling local aspects of a self-reconfiguration process that requires precise module movements.

2.5 Conclusion

There is a good match between the features of lattice automata and what is desired and possible to implement in self-reconfigurable robots. The rules of lattice automata typically only depend on the state of neighbor automata. On real hardware, this information can easily be obtained through neighbor-to-neighbor infrared communication. The distributed nature of lattice automata is also appealing from a self-reconfigurable robotic point of view since distributed control is a key element of making a self-reconfigurable robot robust.

There are, however, assumptions made in the use of lattice automata that are not easily handled in physical self-reconfigurable robots. One assumption is that unlike cells in a computer simulation, modules do not blink in and out of existence. In fact, in order to turn off one cell and turn on another a physical module has to move from the first to the latter. This is a physical process that easily can take on the order of several seconds. This immediately leads to a few problems e.g. the mutual exclusion problem of modules trying to move into the same cell, the problem of motion constraints

where not only the from and to cells are of interest, but also the cells that the module is physically moving through to get from one cell to another are also important. A final assumption, is that in computer simulations it is assumed that cells agree on directions, e.g. based on implementation in a two or three dimensional array. However, obtaining this information in a self-reconfigurable robots either requires dedicated sensors or a distributed consensus algorithm.

While the above problems have been less of a focus in the self-reconfigurable robot community, maybe because they are not apparent in simulation-studies, the problem of scalability has been a corner stone of the work. The main challenge arises from the combinatorial explosion of configuration neighbourhoods and the practical need to differentiate the active lattice automata rules in different parts of the robot.

As we have discussed in this chapter there are a few solutions to these problems. The first and most basic is to limit the task domain and, in addition, consider tasks where the individual module has a large degree of autonomy. In situations, where the modules start to depend on each other more careful considerations have to be done. Typically, this involves a split where local aspects of the control is performed by lattice automata, but there is a distributed algorithm that handles the global aspects of the problem.

From a lattice automata point of view it may be instructive to develop lattice automata where the underlying assumptions are a better fit for the physical modules of self-reconfigurable robots. From a self-reconfigurable robotic perspective it seems there is still potential in exploring hybrid algorithms with a lattice automata basis and a more globally oriented algorithm. From the point of view of mechatronics it may also be possible to develop modules where breaking the assumptions of lattice automata is less of problem. One avenue of research could be soft modular robots that may be able to squeeze past each other passing through a single cell alleviating the mutual exclusion problem.

In conclusion, the interaction between lattice automata research and research on self-reconfigurable robots has been productive and there is significant potential in exploring how the two can benefit each other further.

References

1. Bojinov, H., Casal, A., Hogg, T.: Emergent structures in modular self-reconfigurable robots. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1734–1741. San Francisco, California (2000)
2. Bojinov, H., Casal, A., Hogg, T.: Multiagent control of self-reconfigurable robots. In: Proceedings of 4th International Conference on MultiAgent Systems, pp. 143–150. Boston, Massachusetts (2000)
3. Butler, Z., Kotay K., Rus, D., Tomita, K.: Cellular automata for decentralized control of self-reconfigurable robots. In Proceedings of IEEE International Conference on Robotics and Automation, Workshop on Modular Self-Reconfigurable Robots. Seoul, Korea (2001)
4. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic de-centralized control for lattice-based self-reconfigurable robots. Int. J. Robot. Res. **23**(9), 919–937 (2004)

5. Butler, Zack, Rus, Daniela: Distributed planning and control for modular robots with unit-compressible modules. Int. J. Robot. Res. **22**(9), 699–715 (2003)
6. Christensen, D.J.: Experiments on fault-tolerant self-reconfiguration and emergent self-repair. Proceedings. Symposium on Artificial Life part of the IEEE Symposium Series on Computational Intelligence, pp. 355–361. Honolulu, Hawaii (2007)
7. Fukuda, T., Kawauchi, Y., Buss, M.: Self organizing robots based on cell structures—CEBOT. In: Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems, pp. 145–150 (1988)
8. Fukuda, T., Nakagawa, S.: Dynamically reconfigurable robotic system. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, pp. 1581–1586 (1988)
9. Fukuda, T., Ueyama, T.: Cellular Robotics and Micro Robotics Systems, vol. 10 of World Scientific Series in Robotics and Autonomous Systems, vol. 10. World Scientific (1994)
10. Murata, S., Kurokawa, H.: Self-Organizing Robots. Springer, New York (2012)
11. Murata, S., Kurokawa, H., Kokaji, S.: Self-assembling machine. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 441–448. San Diego, California (1994)
12. Østergaard, E.H., Lund, H.H.: Evolving control for modular robotic units. Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 886–892. Kobe, Japan (2003)
13. Østergaard, E.H., Lund, H.H.: Distributed cluster walk for the ATRON self-reconfigurable robot. In: Proceedings of 8th Conference on Intelligent Autonomous Systems, pp. 291–298. Amsterdam, Holland (2004)
14. Romanishin, J.W., Gilpin, K., Rus, D.: M-blocks: Momentum-driven, magnetic modular robots. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 8288–4295. Tokyo, Japan (2013)
15. Rosa, M.D., Goldstein, S., Lee, P., Campbell, J., Pillai, P.: Scalable shape sculpting via hole motion: Motion plan- ning in lattice-constrained modular robots. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1462–1468, Orlando (2006)
16. Støy, K.: Controlling self-reconfiguration using cellular automata and gradients. In: Proceedings of 8th International Conference on Intelligent Autonomous Systems, pp. 693–702. Amsterdam, The Netherlands (2004)
17. Stoy, K., Brandt, D.: Efficient enumeration of modular robot configurations and shapes. In: Proceedings of IEEE/RSJ International Conference on Robotics and Intelligent Systems, pp. 4296–4301. Tokyo (2013)
18. Stoy, K., Christensen, D.J., Brandt, D.: Self-Reconfigurable Robots: An Introduction. MIT Press (2010)
19. Varshavskaya, P., Kaelbling, L.P., Rus, D.: Automated design of adaptive controllers for modular robots using reinforcement learning. Int. J. Robot. Res. **27**(3–4), 505–526 (2008)
20. Walter, J., Welch, J., Amato, N.: Concurrent metamorphosis of hexagonal robot chains into simple connected configurations. IEEE Trans. Robot. Autom. **18**(6), 945–956 (2002)

Chapter 3
Modular Reconfigurable Robotic Systems: Lattice Automata

Nick Eckenstein and Mark Yim

Abstract Modular and reconfigurable robots hold the promises of versatility, low cost, and robustness. Many different implementations utilizing lattice structures exist, with varying advantages. We introduce, define terms for, and describe in full several key systems. Comparisons between connection mechanisms are made. We describe some of the software concerns for modular robots, and review the applications of self-assembly and self-repair.

3.1 Introduction

Modular and Reconfigurable Robotic Systems (or MRR systems, as they will be referred to in this chapter) are robotic systems that are made up of many repeated modules that can be rearranged. They can be classified by the underlying structure used to organize a group of modules; *chain* type systems are organized into a chain or tree structure whereas *lattice* type systems are organized on a lattice structure. *Hybrid* systems are those that can switch between these organizations. This chapter will serve to introduce the concepts of MRR systems specifically focusing on lattice and hybrid type MRR systems. More detail on the type descriptions is given in Sect. 3.1.2.

A **modular** robot is defined as one that is "built from several physically independent units that encapsulate some of the complexity of their functionality" [37]. A **reconfigurable** modular robot is defined as one where the module's connectivity can be changed. One example of a lattice type MRR system is shown in Fig. 3.1.

N. Eckenstein (✉) · M. Yim
GRASP Lab and Department of Mechanical Engineering and Applied Mechanics,
University of Pennsylvania, Philadelphia, USA
e-mail: neck@seas.upenn.edu

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_3

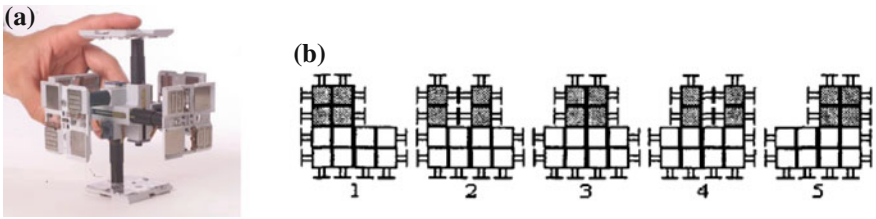


Fig. 3.1 The Telecube [38] system is an archetypical 3D lattice-type modular reconfigurable robot system. The system can expand each side to accomplish motion along the lattice. Docking attachment is accomplished by means of switching permanent magnet faces. The figure on the *right* depicts Telecubes performing a reconfiguration operation [43]. **a** Telecube physical prototype, **b** Telecubes reconfiguring

3.1.1 Motivation

MRR systems hold three promises: versatility, robustness and low cost.

Versatility Typically, a robot is built to certain size and shape specifically for a given application with an associated set of tasks. MRR systems can be reconfigured into many different sizes and shapes and so can achieve many more tasks than fixed configuration systems.

Robustness MRR systems achieve robustness through redundancy and self-repair. If a module or set of modules malfunctions or is damaged, the robot can recognize this and utilize redundant module(s) from elsewhere in the configuration or replace good modules for bad ones.

Low cost MRR systems typically have many repeated modules. This repetition allows batch fabrication and mass production techniques to be utilized to lower the cost of the individual modules.

Note that of these three promises, only versatility has been proven so far. More redundancy means more opportunity for failure, so techniques must be developed that provide graceful degradation of performance with redundancy. Lower individual module cost can help reduce costs, but the overall system cost to achieve tasks is the ultimate goal.

3.1.2 Key Terminology

As mentioned earlier, MRR systems fit into several different types [37, 49]. Examples of these types are shown in Fig. 3.2.

Lattice type systems have modules that are nominally situated on a virtual lattice. Each module occupies one site in the lattice and is capable of moving to a neighboring site. This movement is characterized by a simple motion along a single degree of

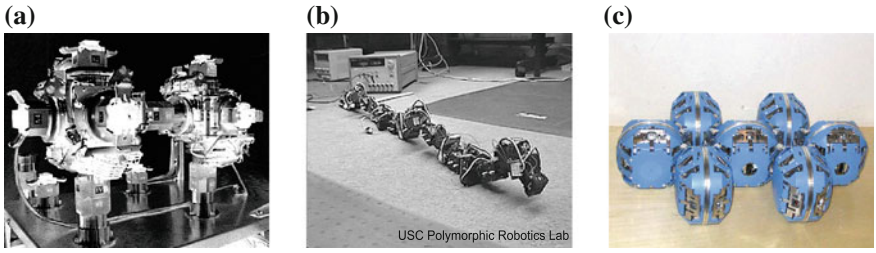


Fig. 3.2 Examples of MRR types. **a** 3D Unit, another lattice-type modular system, **b** CONRO, a chain-type modular robot in a snake configuration, **c** ATRON [27], a hybrid-type robot

freedom (DOF) path with a swept volume local to the neighborhood of the site. The local property of this motion means that the motion planning and collision detection is independent of the number of modules.

Chain type systems are organized in a chain or tree architecture. To reconfigure, chains of modules attach two ends together and break at a different point. Sequences of making and breaking loops serve to reconfigure a system from one shape into a goal shape. For these chains, the computational time complexity of motion planning scales with the number of modules. For this reason, reconfiguration planning, that is the determination of a sequence of motions, attachments and detachments of an MRR system has been much easier with lattice systems. However, chain systems can form articulated arms/legs and tend to be easier to use for robotic tasks such as locomotion or manipulation of objects.

Hybrid type systems can operate as either a lattice type or a chain type system. Their capabilities can be organized either way depending on the application.

Other types for MRR system categorizations [3] focus on reconfiguration mechanisms and include *mobile* type systems. Mobile systems have modules that can move independently in the environment. In this chapter however, the reconfiguration is not as central as the organization of modules, so the mobility of individual modules is listed simply as a property of the system.

Connectedness describes how many faces on one module can be connected to another module. For a given polyhedral base, the number of available connection faces describes the connectedness. For example, a cube has six faces and if all six faces are possible connection faces, the module is said to have full 6-connectedness. It is typically desirable to have full connectedness, but in many cases, it is difficult to implement.

A *configuration* refers to the arrangement of modules with the associated connectivity. Note this does not include the joint angles of the modules which result in a particular shape, or pose of the robot.

Terms for collections of modules include; *cluster*, meaning a small set of contiguous modules, and *meta-module*, which is a configuration of a small number of modules that acts as a repeated element within a larger configuration. Meta-module planning is often used to improve the speed of reconfiguration planning.

3.1.3 Environments

Lattice MRR systems exist in multiple environments, although the primary environment of choice is land or on a pre-existing lattice structure. Land systems will be shown in detail in the section focused on hardware, so here we will note only systems which perform in alternate environments.

Some lattice systems operate with at least one 'anchor site'. This anchor site is either a passive representation of the connector used between modules, or a stationary module others can attach to, serving as a base site and reference frame. Modules which make use of anchor sites include Xbot [46], 3D Fracta [23] and the Crystalline robot [2], all of which use the existing anchor sites for stability and are shown in Sect. 3.3.1.

Recently, the DARPA TEMP system accomplished reconfiguration of on the surface of water, as a testbed for a modular deployable seabase system [26]. Assembly consisted of arranging 33 modules autonomously into a bridge shape, which was then crossed by a remote controlled car. The TEMP system is a more modular version of an earlier project known as the Mobile Operating Base, composed of 3–5 large modules [13]. Other systems in fluid environments include stochastic-based assembly systems from Tolley et al. [39].

The only MRR system to perform in air of note is the Distributed Flight Array, a 2D planar array of rotors which perform decentralized flight control, driving, and docking. In this manner the system can self-assemble, take off, and perform controlled flight [28], though reconfiguration occurs on land.

3.2 Challenges and Practical Issues for MRR

3.2.1 General Limitations

Many limitations exist in the context of MRR systems, principally due to design requirements. Counterintuitively, the repetitive nature of MRR systems can be quite constraining. Required functionality of a system must be decomposed into identical modules, yet having the entire functionality in a single module would defeat the purpose of having and MRR system.

In any MRR system, requirements can be broken into two parts, task requirements and reconfigurability requirements. Since the task is unknown a priori, generic task requirements lead to system characteristics such as strength, size, weight, power capacity, efficiency and precision. Reconfigurability requirements lead to system characteristics involved with attaching and detaching mechanisms.

Development and improvements to an existing design are constrained by the form factor limitations. In a lattice MRR system, the given lattice structure has a characteristic size of a unit cell in the system. For example, the lattice size in the CKBot version 1.0 modules was 60mm $\times$ 60mm $\times$ 60mm. No part of the module may extend beyond this lattice size. The motor in particular often requires a large

footprint within the module, taking up space that could be used for battery, control board, sensors, etc. As with most designs, trade-offs must be taken with the most crucial features having the highest priority of space and position.

To help us further reduce the strain on the lattice space, some implementations of these types of systems move certain function requirements off-board or to separate specialized heterogeneous modules. These functions are then relayed through a tether, bus, or mechanical attachment to the full system. Functions which have been successfully moved outside of the module itself include power, sensing, communication/control, and even actuation [47], as we will note below.

3.2.2 Key Metrics

Comparing MRR designs is somewhat difficult due to the variations in intended application and design requirements. For example, measuring strength alone is not straight forward since modules with larger lattice sizes will typically contain larger motors. To that end, we must reformulate our metrics of comparison to better understand the advantages between different designs. A leading metric for modular systems is the number of modules in a system. The more modules the system supports, the more complex and interesting behaviors exist for the system to perform. Additionally, larger systems may be able to engage larger forces by parallel actuation.

The record for number of modules simulated is 130^3 (2.2 million), for a cube shaped conglomerate of lattice-type simulated modules 130 to a side [8]. The record for number of physical modules implemented at once in a single robot is held by M-TRAN (Mark III), which had produced 50 modules.

In order to accomplish sophisticated mass behaviors with many modules, we wish to have smaller modules—so small size is another desirable quantity. Externally controlled assembly systems have been built as small as $500 \times 500 \times 30 \mu$m employed by Lipson et. al. [39]. Self-actuated systems such as Smart Blocks [31] and Milli-Motein [19] are 10mm in size. With many modules on a significantly small scale, we increase the resolution of our systems and come closer to presenting a seemingly 'continuous' set of behaviors for locomotion, reconfiguration, etc.

Larger modules can be desirable as well—if for example we wish to construct a large structures with a minimum of materials. The Giant Helium Catoms (GHC) currently hold the record of largest module at a cube size approximately 1.9 m on a side.

A key metric to the reconfigurability requirements is the bonding strength of the attaching mechanism. For many systems which uses hooks or latches, the material strength of the hook or latch is the limiting factor and is typically large compared to the strength of actuators. For systems which use magnetic or electrostatic bonding, the strength of the bond is much weaker and can be a limiting factor in the size of a conglomerate system.

A related metric is the force required to un-bond or de-dock. Typically stronger bonds also require concomitantly large forces or energy to undo them. For example,

low temperature melting point alloys can be used to bond two modules, but require a large amount of energy to melt the bond.

3.2.3 Modular Robot Morphology—Shape and Connectedness

MRR Systems have had a variety of shapes and lattice types since the first system was created in 1989 [9]. Shapes that have the properties of being space-filling and easily calculated lattice positions are desirable. With the exception of the cube, most regular polyhedra do not tile space. Two other polyhedra that tile space include a rhombic dodecahedron and a right angle tetrahedron. This tiling implies the underlying lattice structure. For example, the rhombic dodecahedron is the resulting shell of one cell of the Voronoi diagram of the centers of a face centered cubic lattice structure.

Connectedness impacts the range of configurations possible with a given MRR system. Connectedness affects the available graph representations [18] and practical applications—i.e. a cubic system which connects four out of six faces cannot always represent all possible configurations. Most systems also construct the connector with symmetries in such a way that two connected modules can have more than one way to attach. In the M-TRAN system for example, any two eligible faces (that is, a male-female pair) can connect in up to four ways, each option representing a 90° change in orientation.

We will call the combination of the external shape occupied by the module (e.g. the angles of joints in the robot) and the configuration the *morphology* for the purposes of this chapter.

3.3 Example Lattice System Hardware

Here we profile several systems, and discuss relevant features compared across platforms. Each system presented represents a set of solutions to the unique design challenges faced in MRR systems design. We will first present each system in detail and then compare features as they were implemented. Systems are sorted roughly by lattice type.

3.3.1 Key Designs

3.3.1.1 Three-Dimensional Systems

The majority of implementations use cubic shaped modules. Early systems include the 3d-Unit, Molecule and Telecube systems.

The **3d-Unit** (or 3D Fracta, as it is sometimes called) exists on the cubic lattice with full 6-connectedness. The system uses a single motor and a clutch to individually actuate one of the 12 degrees of freedom as required (six for rotation of faces, six for connecting faces) [23]. The 3d-Unit system can be seen in Fig. 3.2a. The connectors are paired at 90° by a special handshake mechanism, meaning a successful connection may require rotation along a face. Power and communication are transmitted through this connection mechanism. Movement of a module in the lattice requires rotation by an adjacent module using a connection on that adjacent module on the axis of rotation. In this way modules can be moved laterally or vertically one position at a time. Actuation for both the face rotation and connection mechanism is accomplished by means of a single DC motor/harmonic drive combination for each module. A diagram of the motion pattern is shown in Fig. 3.3.

The Self-Reconfiguring Robotic Molecule [20], often referred to simply as the **Molecule** system also exists on the cubic lattice. Each molecule is composed of two 'atoms', connected by a right-angle bracket, so each contiguous module takes up three positions on the cubic lattice, in an 'L' shape. Each atom has five connectors to connect to other modules and two actuated degrees of freedom; one about the right-angle connection and one about a single connector. RC servomotors are used for the two rotational degrees of freedom. The connection is accomplished by means of oppositely polarized 1″ electromagnet faces, with an interlocking sheath pattern to prevent undesired rotations. Electronic hardware on the module is composed of a microprocessor and controllers for the electromagnets and servos, with high-level control of the system accomplished by a workstation connected to the module by RS-485 connection. Despite the somewhat strange shape of the module, it has no problems traversing the lattice in a straight line or over convex/concave edges, computing the straightest path in O(n) time. A Molecule prototype can be seen in Fig. 3.4.

The **Telecube** system (Fig. 3.1), like 3d-Unit, exists on the cubic lattice with full 6-connectedness. Each face has a prismatic DOF allowing any side to expand to more than twice its original length. Careful control of connections and use of the prismatic DOFs results in motion in the desired direction along the lattice. This method of module motion is a 3D extension of a 2D system called *Crystalline* [2]. Connection between faces is accomplished by means of permanent magnet faces. These perma-

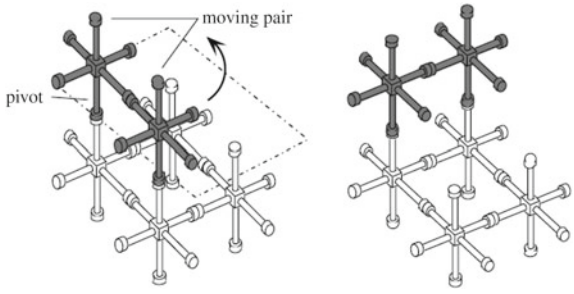


Fig. 3.3 3d Unit Lattice motion pattern

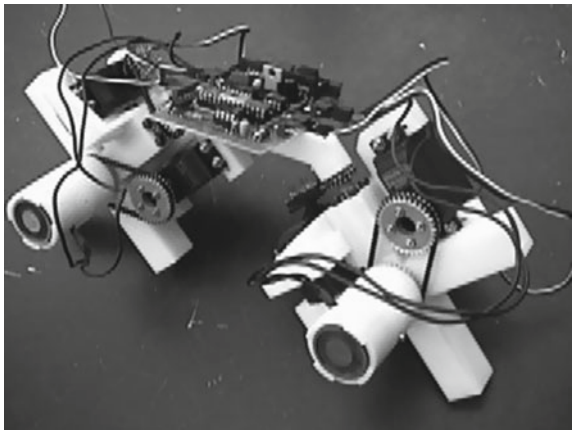


Fig. 3.4 Molecule module prototype. Note the two 5-connected 'atoms', and the *right-angle* bracket which takes up a lattice position. *Photo* Keith Kotay

nent magnet faces have two layers of neodymium magnets, and connection status is changed by moving one set of magnets one pitch length thus changing the path of magnetic flux. The magnets are moved by a shape memory alloy (SMA) mechanism which shifts the layer. The frame material is chosen both because it is light and because it is internally lubricated to provide low friction. The linear actuation is achieved via brushless DC motors attached to worm gears.

The largest MRR system in the literature is the **Giant Helium Catoms** (**GHC**), developed at CMU. One possible application is extraplanetary structures where ultralight expandable modules would be useful for structural applications [17]. The GHC exists on the cubic lattice with full 6-connectedness. Connection between modules is accomplished by use of a novel electrostatic adhesion mechanism. Each face has four flaps which contain two electrodes each, with a dielectric material (mylar) in between. This allows charges to build up across the module interface, creating the electrostatic attraction. Each flap corresponds to an edge on the face, as seen in Fig. 3.5. The flaps themselves can be actuated to extend by means of Nitinol (SMA) wires, with reverse actuation by a constant force spring to close the flap back down. Each module was filled with helium, allowing for a total module weight of 50 g, despite the modules being approx. 1.9 m on a side. Each module had a central processing board and six outer slave boards, one for each face, as well as its own battery for power and Zigbee for wireless communication, although power transmission is proposed between modules via the adhesion interface. Sensors and actuators are controlled using the I^2C bus. Each flap angle was controlled and sensed by a combination of the flap voltage and a potentiometer to measure angle.

The **M-TRAN** system (Mark III) [21] is a hybrid system that can be organized on a simple cubic lattice (Figs. 3.6 and 3.7). Each module is composed of two cubes with a revolute DOF about the center of each cube relative to a link that attaches to both. The blocks each occupy a single simple cubic lattice site; thus each module occupies two adjacent sites. As a hybrid system the modules can form chains to perform articulated tasks but can also arrange themselves on the cubic lattice for

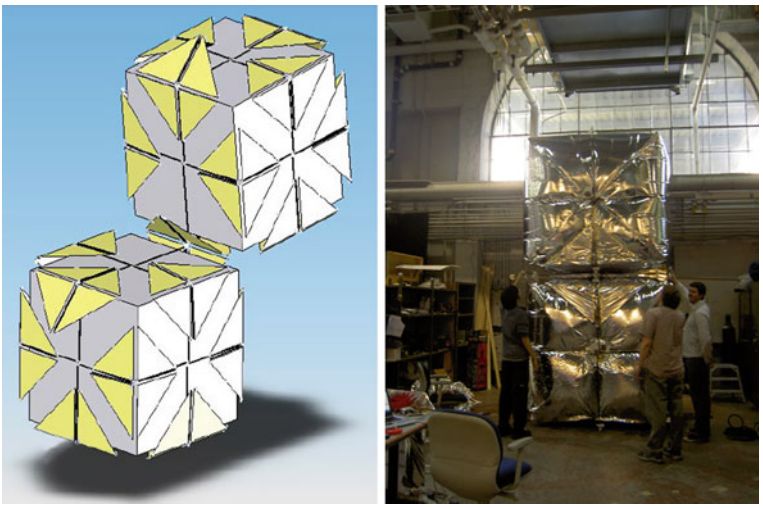


Fig. 3.5 GHC robot with 4 flaps on each face, 24 flaps total. Each flap can be rotated about the edge to accomplish module motion, as seen in the *left-hand* image. The image on the *right* shows two prototypes stacked on *top* of one another

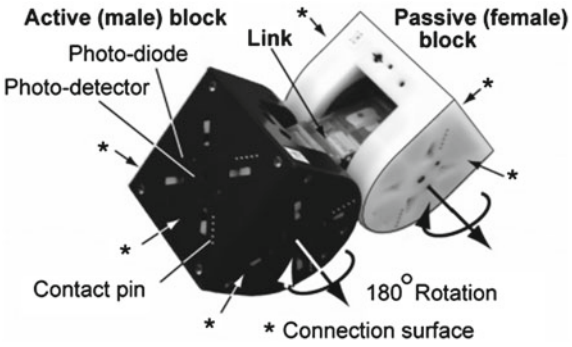


Fig. 3.6 M-TRAN III Module

reconfiguration. One block has three active latching male connection mechanisms; the other has three passive female connection plates. Each module has a total of five DC motors (two for the link section—one for each block, plus one for each of the three active male faces). Module motion is accomplished by the combined motion of the two link motors, as well as selective attachment/detachment to other modules in the lattice. The conjoined male/female combination of modules results in a tiling alternating block gender along each cartesian coordinate. It checkerboards the a three-dimensional space. Each connector pair can be oriented at an offset of 0, 90, 180, or 270°, giving four symmetric possible orientations. By clever arrangement of modules then it is possible to move modules from one axis-alignment to another during reconfiguration. M-TRAN III carries the distinction of having demonstrated

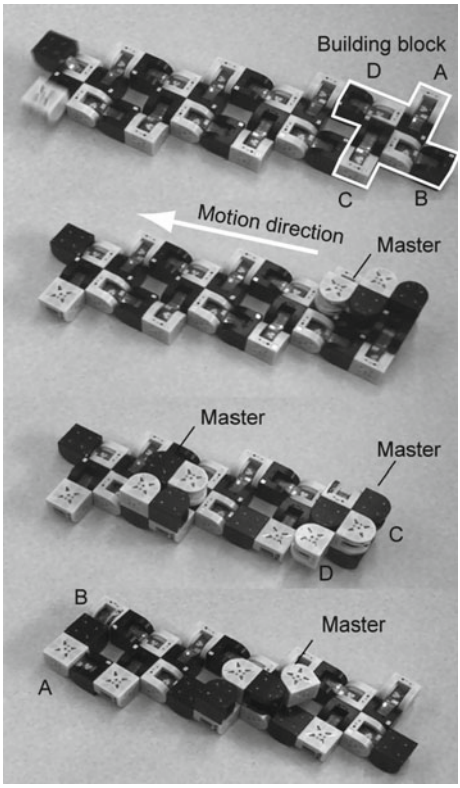


Fig. 3.7 M-TRAN Mark III moving along a lattice structure by reconfiguration

the most reliability of a system of its complexity, having changed surface connections by up to 24 modules over 100 times.

The **SMORES** modular system [5] are organized on the cubic lattice, forming a cube with four connection ports. It has four actuated DOF. Two of the DOF serve as wheels allowing each module to move independently, making SMORES a mobile system. The other two actuation degrees of freedom allow tilt and pan of the module faces. This design has the ability to emulate many other types of modular robots and progress towards a "universal" modular robot. In this way the SMORES system is capable of emulating many of the existing lattice and chain type modular robots successfully, in the hopes that the system will have the combined capabilities of many of the existing systems.

A recent system developed at MIT, **M-Blocks** (Fig. 3.8), exists on the cubic lattice with full 6-connectedness [33]. It uses internal inertial exchange to move modules in the lattice. A flywheel located within the module in combination with a belt brake allow the module to abruptly exchange angular momentum. The external frame contains of a set of 24 diametrically magnetized magnets (2 on each edge). These magnets ensure that edge-edge contact is maintained during the motion, with another set of 8 smaller magnets on the faces to ensure the module bonds in an aligned position.

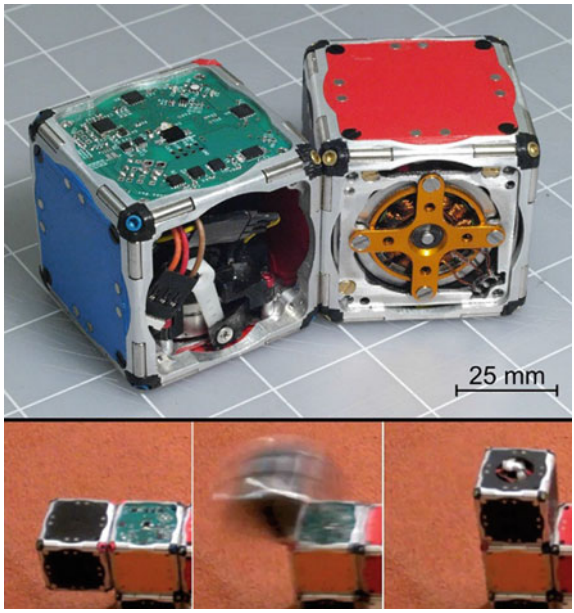


Fig. 3.8 M-Blocks robots. *Top figure* shows the hardware, including the magnet patterns on the edges/faces, as well as the flywheel. *Bottom figures* show the path taken by a module during a typical convex transition pivot motion. *Photo* by John Romanishin (Daniela Rus's Distributed Robotics Laboratory at CSAIL, MIT)

Presently the system can only move in the plane perpendicular to the axis of rotation of the flywheel, though developing a system with three perpendicular flywheels is feasible. Due to the unique mechanism for motion, modules are capable of moving independently of the lattice, making them a *mobile* system. Each module also contains its own power, wireless (Xbee) communication, and a 32-bit /ARM microprocessor. Sensors include a 6-axis IMU, an IR LED/photodiode pair for intermodule communication, and Hall Effect sensors to detect misalignments. Reconfiguration planning requires a bit of compensation for the way in which the modules pivot about an edge, precluding other modules from occupying corresponding positions which could block the motion. The authors address these issues with a Pivoting Cube Model (PCM) for reconfiguration.

ATRON, a system developed at the University of Southern Denmark, is the only modular system modeled on a "face-centered cubic" lattice, allowing connections to up to 8 other modules [27]. Its major components can be seen in Fig. 3.10. The module has two halves which can spin relative to one another somewhat like a wheel. This motion won't cause the module itself to move itself to another lattice position, but it will move two other modules relative to each other. Each half of each module has two actively-driven male connectors and two passive female connectors, as well as a microprocessor. Each module carries its own power, and the two halves of the module share power and communications through a large slip ring in the central plane of the module. This allows for continuous uninterrupted motion of one half relative to the other, for 'wheel' type functionality in a module.

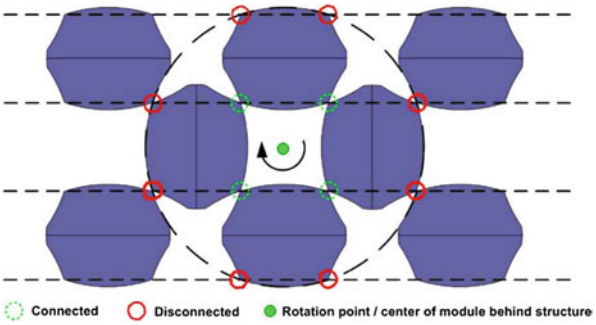


Fig. 3.9 ATRON Robot, performing swap with a full lattice. The *center* point represents an out-of-plane module which rotates to perform the swap

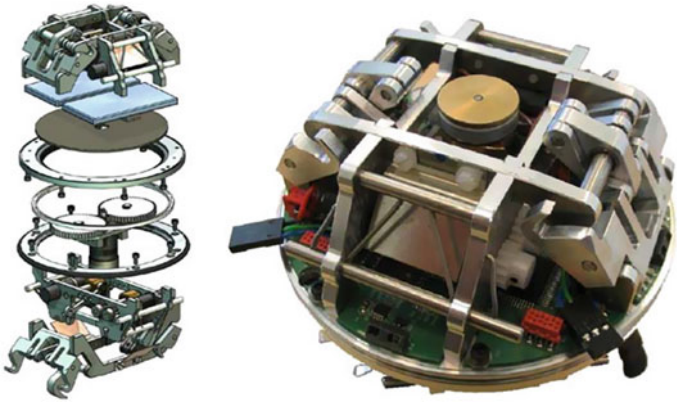


Fig. 3.10 ATRON Robot, Exploded view and full prototype without plastic cover

Inter-module connection is accomplished by mechanical arms which reach out and 'grab' the passive connector, simultaneously aligning it and making a solid mechanical connection. Inter-module communication is accomplished by an infrared emitter-detector pair which also serves as a proximity sensor. The lattice choice combined with appropriate shaping of the module exterior permits modules to be moved even in a fully-packed lattice, as shown in Fig. 3.9. ATRON can connect its modules only orthogonally (that is, at the 90° angles seen in Fig. 3.2c), and so has no orientation options between two modules like other systems. With the large hooks for latching, the connection system has one of the strongest bonds, but also consumes a majority of the space within each module.

3.3.1.2 Two-Dimensional Systems

Although full-scale reconfiguration would ideally be on a three-dimensional lattice, many two-dimensional systems have made interesting advances in the technology. In

particular, removing the necessity to compensate for gravity reduces the functional strength requirement, allowing experimentation with alternative actuation methods and reducing actuation overhead. The following systems are organized on a two-dimensional lattice.

The **X-Bots** system developed at the University of Pennsylvania [46] is organized on the 2D lattice with full 4-connectedness. Each X-bot is simplified by containing only local power, processing and connection systems. Communication is performed by conductor contact at the connection points. Actuation is located externally, by placing the system on an X–Y stage. In this way the system can use selective connection in combination with inertial motion to reconfigure the system, as shown in Fig. 3.11. In addition to rotations of a single module around an axis, by selective disconnection the system has demonstrated more complicated two-module 'motion primitives' to enable reconfiguration into arbitrary conglomerate shapes. As with many systems, a proof is shown that any arbitrary shape can be obtained. An algorithm is developed that determines a sequence of motions that transform any configuration into a canonical configuration (e.g. a single line of modules). This sequence is reversible so any configuration can transformed into any other by transforming into the canonical configuration and reversing the sequence into the goal configuration.

The **Micro Unit** system [53] exists on the 2D lattice with full 4-connectedness. Each module has two male and two female connectors at the corners, about which the modules rotate. All rotation and actuation of the latching for connection is accomplished by SMA wires heated electrically. These wires allow rotation at the corners between modules, as well as activation/deactivation of latches between modules. The Micro Unit system is one of the smallest systems prototyped at a system size of 2 cm. A prototype can be seen in Fig. 3.12.

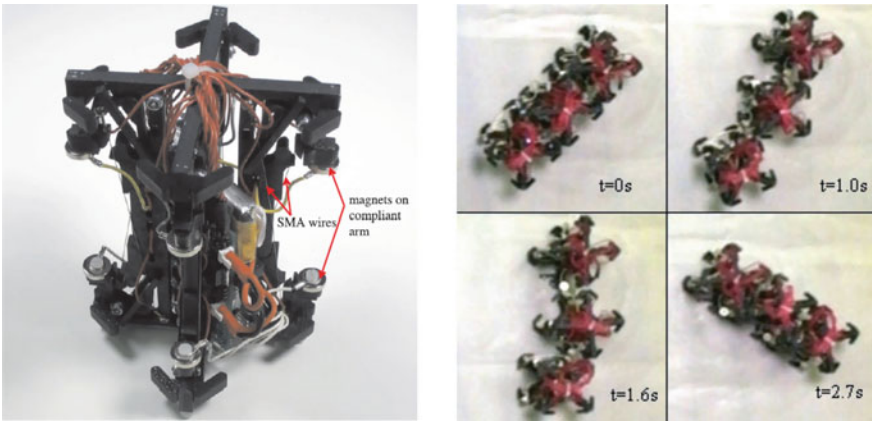


Fig. 3.11 *Left* X-Bots module. Connection magnets and SMA wires are visible, power and processing contained within frame. *Right* X-Bots module undergoing complex motion primitive. By disconnecting two modules at specific points, the inertial motion can reconfigure both at once

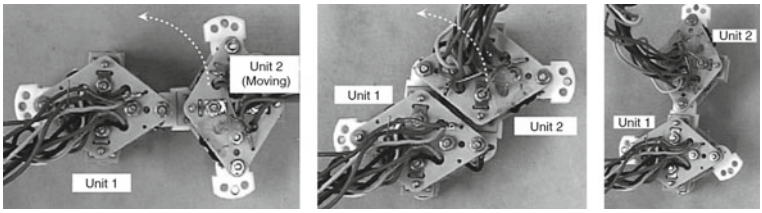


Fig. 3.12 The Micro unit system undergoing reconfiguration. Male and female connectors are visible

The 1-cm **Pebbles** system, developed at MIT, was the first to make use of electropermanent magnets as a connection mechanism for modules. These devices consist of two rods of different types of magnet materials with nearly the same magnetic strength but widely differing coercivity, capped with iron and wrapped in an electromagnet coil. The two rods are made of Neodymium-Iron-Boron (NdFeB) and Alnico V, respectively. The Alnico magnet switches its polarity much more readily, so a pulse from the electromagnet coil switches this magnet, but not the NdFeB. If the two magnets have the same polarity, magnetic flux points outward and the module can attract other modules. If however, the Alnico is flipped by the coil and has the opposite polarity of the NdFeB magnet, the flux circulates within the EP magnet and does not leave through the poles. While this mechanism requires some power to change the polarity of the Alnico magnet and switch states, it does not require any power once a state has been set; it is bistable. Power is transmitted between modules by conductor contact and communication is transmitted by induction through the magnets. Module motion was not implemented for this system; the idea is for construction of a shape by *self-disassembly*, rather than self-assembly or self-reconfiguration. This means shapes are formed by deactivation of the EP magnets, allowing extraneous modules to drop out when external force is applied to the system. You can see several shapes formed by Pebbles in this way in Fig. 3.13, along with a prototype.

The **EM-Cube** [1], presented in 2008 by An, exists on a 2D square lattice, with full 4-connectedness. Each module contains a microprocessor and a Zigbee chip for wireless communication. Power is supplied externally. The motion method for the EM-Cube is novel; two faces (bottom and left) contain a pair of permanent magnets. The other two faces (top and right) contain three electromagnets. By changing the polarity of these electromagnets, the overall magnetic force changes, allowing the EM-Cube to move through a five-step process from one module to the next, as seen on the right in Fig. 3.14. So long as all modules are placed in the lattice with the same orientation, any module can be moved—either with its own electromagnets or by the neighboring modules. However, some creativity is required for a module to move around a convex corner, as you can see in Fig. 3.15. Since a surface of two modules is required to move a module, two modules must move together to accomplish the convex transition. An also presents other motion algorithms for four-module conglomerations, including one that accomplishes motion as long as it is allowed to run, automatically accounting for convex/concave transitions.

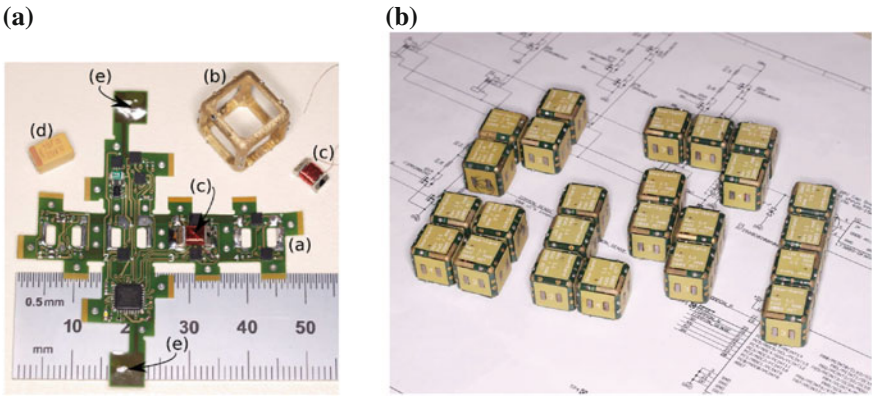


Fig. 3.13 Pebbles system. Photos by Kyle Gilpin (Daniela Rus's Distributed Robotics Laboratory at CSAIL, MIT). **a** Pebbles module, with flex circuit exposed. 4 sites for EP magnet placement are visible. A fully constructed module fits within the cube frame, **b** Pebbles arranged into a variety of 2D shapes [12]

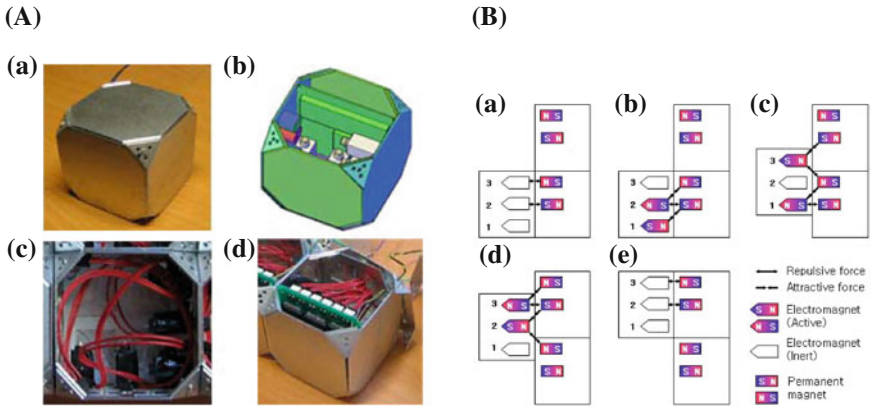


Fig. 3.14 EM-Cube System. [1]. **a** EM-Cube protype. Note in (**c**) that the electromagnets are only on two faces, **b** EM-Cube sequence of magnet switching for motion. The combination of repulsive and attractive forces at each step results in net motion

3.3.2 Lattice Locomotion

Many different types of lattice locomotion exist—each system seemingly has its own motion primitives capable of moving a module from one position in the lattice to another.

There are two types of module locomotion in MRR systems—pivoting (rotational) and sliding (translational). Rotational motion is easier to accomplish due to the ability to use standard motors without a linear drive mechanism, saving space. Typically the center of this rotation is at or near the center of a module connected adjacent to the

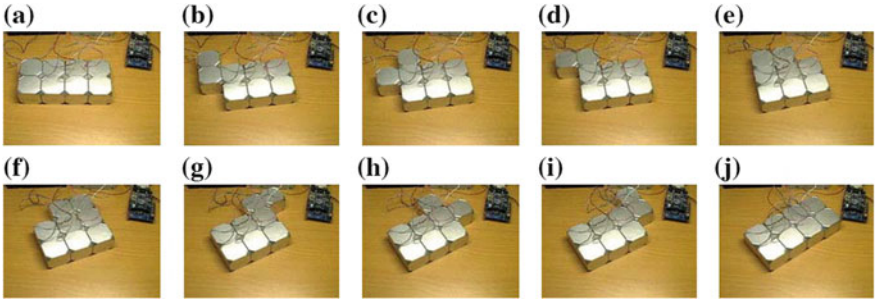


Fig. 3.15 EM-Cube undergoing a convex transition. More than one module is required to maintain a surface against which the module can move

moving module, but this is not always the case. Corners and edges are sometimes used as 'pivot points' to stabilize otherwise unstable motions, resulting in slightly different centers of rotation. Depending on the center of rotation, pivoting modules have some 'exclusion zone', where other modules cannot be located if the module locomotion action is to be successful. Pivoting systems with an exclusion zone thus have fewer locomotion options than sliding modules. Most systems are of the pivoting type. Sliding type systems are less common but include systems such as Crystalline [2], Telecube [43], EM-Cube [1], and Smart Blocks [31].

In pivoting type modules with only one motor (or multiple motors with the same axis) some limitation in locomotion results. For example, if all the modules in a configuration have the same axis of rotation they will be unable to leave the relevant plane, even if they otherwise exist on the three-dimensional lattice. So in systems such as M-TRAN [21] and CKBot [51], care must be taken to add sufficient modules of different axes to permit full reconfiguration capabilities.

3.3.2.1 Locomotion Actuators

Actuation in MRR is typically performed by traditional electric motors, or servomotors. These have the advantage of being relatively easy to use, having easy power transmission, and high strength. However, they have a tendency to take up a lot of space and do not scale well. In particular, scaling down electric motors quickly leads to a significant loss of strength. Functionality such as self-reconfiguration requires additional actuators for attaching/detaching, increasing the importance of actuation in platforms that self-reconfigure.

As a result, alternative actuation has been studied extensively for MRRs. Magnetic bonding methods are attractive due to their self-aligning properties. Standard electromagnets require very large currents to generate enough attraction or repulsion power and are not practical for battery powered MRR systems. Switchable permanent magnets and electropermanent magnets scale relatively well, and are utilized to perform attachment/detachment with a relatively low design burden. These

magnets make use of a permanent magnet-electromagnet pair to change the attraction behavior. Telecube and Pebbles both use this technology for face-to-face attachment [11, 43]. Telecube uses a physical SMA mechanism to move the magnets while Pebbles use electropermanent magnets. Electropermanent magnets can be electronically 'switched' off or on by application of the magnetic field from an electromagnet. Recently, an electropermanent magnet 'wobble' motor has been designed for the Milli-moteins system allowing for useful actuation at the 1-cm scale [19].

Another alternative actuation method proposed for modular robots uses active materials such as SMA [53] or DEA [48] (dielectric elastomer actuation) to perform the primary motion of the modules. These methods show promise for scalability, but can have other drawbacks; SMA is slow to respond and is not very consistent due to its dependence on ambient temperature, and DEA requires thousands of volts with reliability and robustness issues.

Incomplete actuation and external actuation is also presented in some types of systems, as we show in Sect. 3.5.1. These solutions are useful by giving up space inside a module for other components.

3.3.3 Connection Types

Essential to the act of reconfiguration (whether self-reconfigured or not) is the mechanism by which modules are physically mated together. There are many different ways to characterize these connection mechanisms. Table 3.1 shows many of the MRR systems and their connection mechanisms.

Several terms used here to categorize these connectors are explained below.

Self-Aligning Degree represents the degree to which the connector *passively* aligns the two faces, such as magnetic or mechanical forces. A 'High' Rating indicates self-alignment capability in one offset direction approximately greater than 20 % of the characteristic size of the module face. 'Low' rated systems have some self-alignment capability but less than 20 %. 'None' rated systems have no self-alignment capabilities and must be aligned carefully either by active robotic mechanism or by hand.

Gendering represents whether connectors are interchangeable or must be paired in a particular manner. Gendered connectors have a 'male' and 'female' face—male faces can only pair with female faces, and vice versa. Ungendered connectors do not have this restriction—any face can pair with any other face.

Connection Activity and **Disconnection Activity** indicate whether the act of connection/disconnection requires an action on the part of a module. **Connection Agency** and **Disconnection Agency** indicate which modules are required to be operational/active for the respective action. Double End Agency requires both faces to cooperate to accomplish the connection/disconnection, Single End Agency requires only one functioning face (either one), and Male/Female requires the indicated (single) face.

Table 3.1 Connector systems classification; various systems by year

Connector	Month/ Year	Self-aligning degree	Gendering	Connection activity	Disconnection activity	Connection agency	Dis connection agency	Connection type	Connection maintenance	Compliance	Approach angle
CEBOT Mark II [9]	May/ 1989	High	Gendered	Active	Active	Male end	Male end	Latching	Zero-power	Rigid	Perpendicular
Fracta 2D [22]	May 1994	High	Gendered	Active	Passive	Male end	Male end	Magnet	Powered	Compliant	Non-perpendicular
Meta-morphic [29]	August 1996	High	Gendered	Active	Active	Female end	Female end	Latching	Zero-power	Rigid	Non-perpendicular
3-D Unit [23]	May 1998	Low	Ungendered	Active	Active	Double end	Double end	Latching	Zero-power	Rigid	Perpendicular
Molecules [20]	May 1998	Low	Gendered	Active	Passive	Double end	Single end	Magnetic	Powered	Compliant	Perpendicular
Telecubes [38]	May 1998	Low	Ungendered	Passive	Active	Double end	Single end	Magnetic	Zero-power	Compliant	Perpendicular
I-Cubes [41]	Apr 2000	None	Gendered	Active	Active	Double end	Double end	Latching	Zero-power	Rigid	Perpendicular
CONRO [4]	Nov 2001	Low	Gendered	Passive	Active	Female end	Female end	Latching	Zero-power	Rigid	Perpendicular
DRAGON [25]	Dec 2002	High	Ungendered	Passive	Active	Double end	Double end	Latching	Zero-power	Rigid	Perpendicular
Polybot G3 [50]	March 2003	Low	Ungendered	Passive	Active	Double end	Double end	Latching	Zero-power	Rigid	Perpendicular
Crystalline [2]	Sep 2003	Low	Gendered	Active	Active	Male end	Male end	Latching	Zero-power	Rigid	Non-perpendicular

(continued)

Table 3.1 (continued)

ATRON [27]	Oct 2004	None	Gendered	Active	Active	Male end	Male end	Latching	Zero-power	Rigid	Non-perpendicular
amour [42]	Apr 2005	High	Gendered	Active	Active	Female end	Female end	Latching	Zero-power	Rigid	Perpendicular
Slimebot [35]	Apr 2005	None	Gendered	Passive	Passive	Single end	Single end	Velcro	Zero-power	Compliant	Perpendicular
GHC (Catom) [17]	May 2005	None	Ungendered	Active	Passive	Double end	Double end	Electro-static	Low power	Compliant	Non-perpendicular
Molecubes [54]	August 2006	Low	Ungendered	Passive	Active	Single end	Single end	Magnetic	Zero-power	Compliant	Perpendicular
CKBot (SAE) [51]	Oct 2007	Active	Ungendered	Passive	Manual	Double end	Double end	Magnetic	Zero-power	Compliant	Perpendicular
M-TRAN III [21]	Mar 2008	None	Gendered	Active	Active	Male end	Male end	Latching	Zero-power	Rigid	Perpendicular
Roombots [36]	May 2008	None	Ungendered	Active	Active	Male end	Male end	Latching	Zero-Power	Rigid	Non-perpendicular
SINGO [34]	May 2009	High	Ungendered	Active	Active	Single end	Single end	Latching	Zero-power	Rigid	Perpendicular
RATChET [47]	Sep 2009	Low	Gendered	Active	Passive	Single end	–	Magnetic	Zero-power	Rigid	Non-perpendicular
Pebbles [11]	May 2010	None	Ungendered	N/A	Active	N/A	Single end	Magnetic	Zero-power	Compliant	N/A

(continued)

Table 3.1 (continued)

Vacuubes [10]	May 2010	High	Ungendered	Active	Passive	Single end	Single end	Air Pressure	Single pump per cluster	Rigid	Perpendicular
DFA [28]	Sep 2011	High	Ungendered	Passive	Passive	–	–	Magnetic	Zero-power	Rigid	Perpendicular
SMORES [5]	Oct 2012	Low	Gendered	Active	Active	Male end	Male end	Mag/ Latching	Zero-power	Rigid	Perpendicular
M-Blocks [33]	Nov 2013	Low	Ungendered	Passive	Passive	–	–	Magnetic	Zero-power	Compliant	Non-perpendicular
DARPA TEMP [26]	June 2014	High	Gendered	Active	Active	Double end	Double end	Latching	Zero-power	Compliant	Perpendicular

Table 3.2 Table of ZRAA metrics, normalized relative to characteristic length of the face

System	Normalized ZRAA sum
GENFA connector	0.00353
Polybot	0.00503
M-TRAN III	0.00592
JHU	0.00592
I-Cubes*	0.0187
CONRO*	0.0425
Vacuubes	0.0555
ACOR(unpaired)	0.0711
SINGO Connector	0.306
DRAGON	0.353
amour	1.57
3D X-Face	2.00

These are exact where possible from the data available in the literature, otherwise estimated. Entries marked with a * indicate estimated ZRAA rather than exact. Table reproduced from [7]

Connection Type indicates the mechanism by which connections are accomplished. Most systems use either magnetic mechanisms or mechanical latching with a few systems using electrostatic forces or pressure to maintain the connection.

Connection Maintenance indicates the extent to which power is required to maintain a connection. Generally speaking, it is undesirable to have a system require power simply to maintain its shape. This is especially true in modular systems which often have a limited power budget.

Compliance indicates the flexibility of the connection. 'Rigid' connections have a mechanically rigid connection between module frames. 'Compliant' systems have some flexibility to external forces, either from springs/compliant parts or magnetic compliance.

Approach Angle indicates the direction of approach that the system most regularly encounters. Systems with a direction of approach perpendicular to the face are generally more responsive to self-alignment design features.

One metric by which connectors are measured is known as **Area of Acceptance**. Area of acceptance is defined as "the range of possible starting conditions for which mating will be successful" [6]. Practically speaking, what this means is that if the docking procedure is executed given some initial misalignment offset between connectors, the alignment features of the connector will correct the offset. The range over which this occurs is the area of acceptance. Area of acceptance can be difficult to determine; for three-dimensional systems it contains two positional offsets and three orientation offsets (we consider all points along the approach direction to be the same, removing one translational DOF). The concept of Zero Rotation Area of Acceptance (ZRAA) is one simplification which assumes that all orientation degrees of freedom are removed and the approach direction is perpendicular to the face. For purely mechanical self-alignment (e.g. no magnets) a set of active and passive connectors from the literature is characterized in Table 3.2 as a sum of the positions normalized with respect to the connector cross-sectional area.

The concept of area of acceptance is important because alignment and connection systems need to be error-tolerant in order to be successful. Long module chains in particular have a tendency to accumulate errors quickly resulting in failed connections if connectors are not sufficiently corrective. These errors could be in multiple dimensions at once, so it is best to measure the area of acceptance over as many dimensions as is feasible.

3.4 Software Systems for MRR

Software for modular robots poses some interesting problems. Since the robots themselves are typically not that complex, dynamic control issues are not generally discussed, although in some instances high accuracy in position and controllability is desirable. The relevant problem for lattice reconfiguration is at the system level, planning for the reconfiguration of the system in a failure-proof and distributed way. Since MRR systems do not always have a centralized controller, planning and issuing commands, decentralized planning algorithms become necessary.

3.4.1 Reconfiguration Planning

In addition to the typical collision-free motion planning problems that exist throughout robotics, MRR systems have a separate category of planning problem, called **reconfiguration planning**. These problems require the system to recognize its configuration and then find a sequence of reconfigurations to reach a goal configuration. The reconfiguration planning problem does not deal with the specific path or dynamics of the system; rather it is a sequence of viable configuration changes from the initial configuration to the goal configuration. These configurations can be represented in the literature as a diagram or graph of connected modules.

If the robot is not explicitly given its initial configuration, *configuration recognition* is a critical step. The robot or system must first identify the current configuration before reconfiguration can occur. This requires the ability to sense neighbors and can be done in a decentralized [30] or centralized [26] manner.

Once the initial configuration is determined, a *reconfiguration plan* must be calculated. Doing this in the smallest number of moves has been shown to be an NP-complete problem [14]. Reconfiguration planning is largely dependent on each particular system and its design. In particular the lattice type, connectedness, method of locomotion, torque limit, and exclusion rules due to method of locomotion all contribute to the set of rules that define the reconfiguration problem. Thus each system design typically has had specific algorithms to most effectively find a reconfiguration plan; for the Metamorphic system, a heuristic based on Simulated Annealing [29], for the M-TRAN system, a centralized two-layer planner (first with locomotion by meta-module, then locally cooperative behavior rules) [52]. The DARPA TEMP

system converts the goal configuration to a directed graph and grows the assembly outward from a chosen 'seed' module [26]. The methods are nearly as varied as the systems, and typically are developed to best fit the individual system with a quickly computed solution, or more optimally a quickly executed one.

3.5 Assembly Robotics

A common application for MRR systems is assembly. This assembly can be directed either externally to the formation of a superstructure such as a truss or other object, or internally to the assembly of robots from modules available. These tasks however have fundamental considerations in common such as the availability of materials, transport of materials, assembly order, etc. It is also often desirable to have some decentralized method of assembly so that a malfunction of one component does not hinder the system, and so that the system can be reconfigured at will. To this end algorithms have been developed which permit the centralized or decentralized assembly planning and execution.

3.5.1 Self-Assembly and Self-Repair

Thanks to the interchangability aspects of MRR systems, they are capable of performing operations such as self-assembly and self-repair. These operations contribute to the robustness and versatility of MRR systems by allowing for damage and failure scenarios.

Self-repair in modular robots has taken several forms. The intended mechanism is more like self-replacement or self-reassembly; non-functioning modules are abandoned and replaced with a functional module rather than repaired per se. Regardless, this mechanism is highly useful, and relatively less costly the more units exist in the system. The 'Unit' systems (2D, 3D, and Micro Unit) have demonstrated the capability for self-repair by moving defective modules out and replacing them with (previously) redundant modules [24]. This is due in part to the fact that each module is capable of moving a damaged module and disconnecting from it. These are essential qualities of the design for this kind of self-repair since the functions of the defective module cannot be relied upon. An alternate kind of self-repair in mobile clusters occurred with CKBot [51], where the clusters were attached manually and then kicked apart (Fig. 3.16). The clusters were then able to self-right, locate other clusters and cooperatively reassemble.

Self-assembling systems like Molecube [54] and White et. al's systems [45] are capable of creating large structures from very simple modules. Sambot has demonstrated self-assembly [44]. Self-assembly of structural components using an expanding spray foam has been accomplished to support standalone clusters of modules and create a conglomerate robot [32]. A more complete survey of self-assembly in robotic systems can be found in [15].

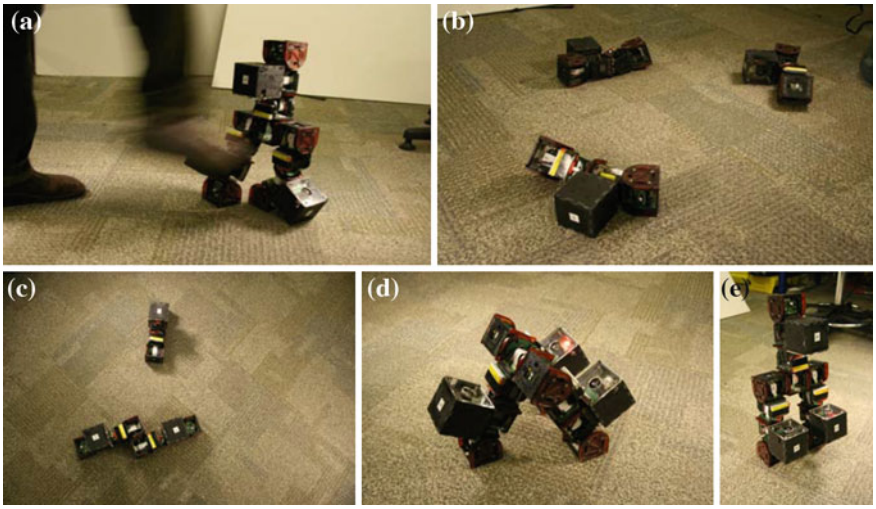


Fig. 3.16 CKBot Self-Reassembly procedure

A term that incorporates self-assembly is **Programmable Matter**. The goal of programmable matter is to have a massive conglomeration of very small mechatronic devices capable of reassembling themselves from one form to another. While the types of systems we have seen already may someday be capable of this kind of application, at present they are too large and make use of technology that is difficult scale down (i.e. electric motors, which lose strength relative to their size very quickly as they scale down). Therefore, alternative systems have been implemented to combat these kinds of problems.

The Pebbles [12] system, mentioned earlier, makes use only of EP magnets which is easier to reduce in size. The X-bots [46] system mentioned above uses inertia to move one or two modules about a lattice at a time. The RATChET system demonstrates how a system can be constructed using a long chain and two external actuators by a combination of inertia and smart design [47]. This implementation allows shapes to be formed from a chain of N modules, while keeping the number of actuators constant, and off-board of the modules. A typical formation sequence is shown in Fig. 3.17. Connection between modules is magnetic, with magnets being released into the 'active' (ready to connect) position by SMA wire. By activating the correct magnets and correctly utilizing external actuation nearly any shape can be formed from a chain of these modules.

Stochastic configuration of passive components on a lattice has been done at several different scales, mostly by Tolley, Lipson, et al. [16, 39, 40]. This is accomplished in a fluidic environment with an array of ports set up to perform as either source or sink. Totally passive mechanical modules with a passive latch are introduced, and then allowed to settle into the area around the desired sink(s), where they latch, reaching the desired assembly before being released. Larger structures require

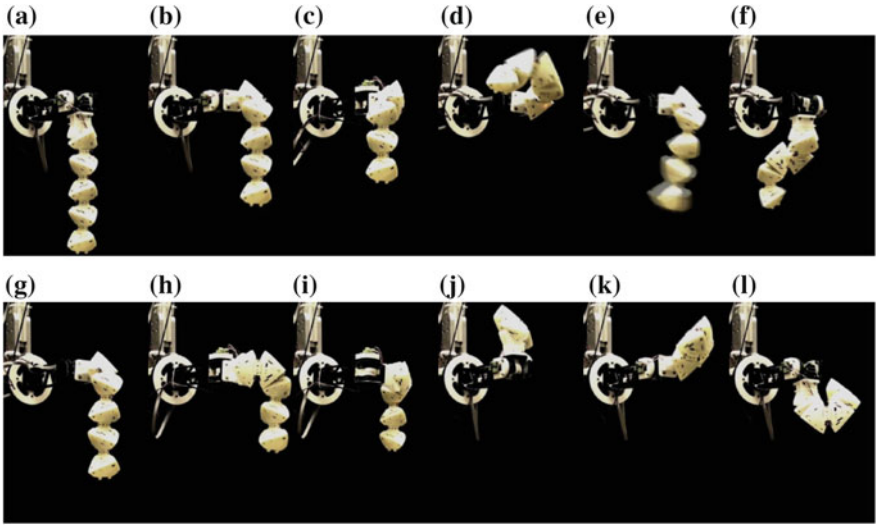


Fig. 3.17 RATChET Chain assembling under external actuation [47]

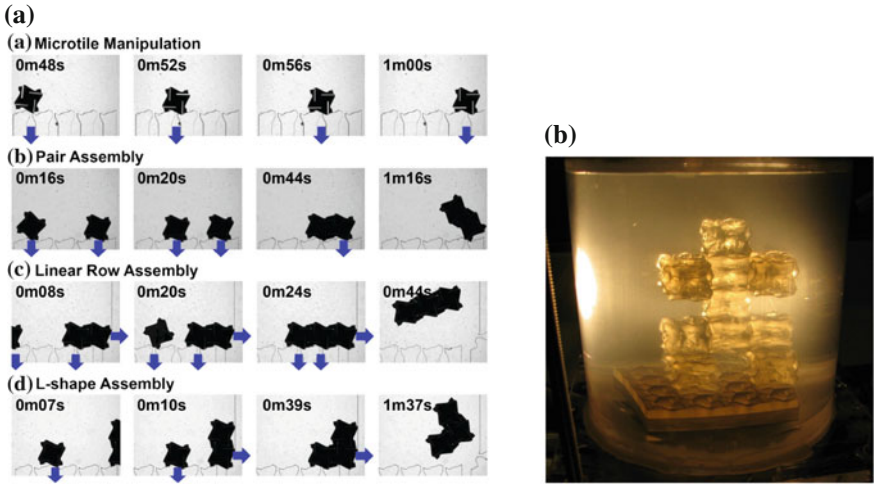


Fig. 3.18 Fluidic Systems **a** Microfluidic components in assembly procedure. Hydrodynamic forces accomplish the relative motion of the modules, with latching beginning a natural mechancial consequence of the shapes being forced together, **b** A three-dimentional fluidicstocgastic system. Each module is 1 cm in size, and is latched mechanically to its neighbors. Array of ports is visible at the *bottom* of the tank

re-trapping an assembly already made in a different orientation so that additional parts can be added. Since different control is required to attract, repel, and latch the modules, visual feedback is required. This means that presently the systems are not autonomous but rely on the input from a human operator. The system also relies

on stochastic motions present in the environment to attract modules, and have them approach in a way that results in alignment. Examples of these systems are shown in Figure 3.18.

3.6 Conclusions and the Future of MRR

MRR systems hold the promise of being versatile, robust and low cost. Several lattice and hybrid systems have been presented in the literature both in 2D and 3D. The lattice structures utilized have mostly been square or cubic but other lattice shapes have shown to be useful as well. These systems assemble, repair, and reconfigure themselves in various ways which enable versatile and robustly functional robotic systems.

In the future, we hope to see MRR systems which become smaller, stronger, and more numerous to enable greater utility. To date there are dozens of groups around the world working on these systems, from both hardware and software points of view. With the continued progress of the research literature the three promises of MRR systems will be seen.

References

1. An, B.K.: Em-cube: cube-shaped, self-reconfigurable robots sliding on structure surfaces. In: IEEE International Conference on Robotics and Automation ICRA 2008, pp. 3149–3155 (2008)
2. Butler, Z., Rus, D.: Distributed planning and control for modular robots with unit-compressible modules. Int. J. Robot. Res. **22**(9), 699–715 (2003)
3. Casal, A., Yim, M.H.: Self-reconfiguration planning for a class of modular robots. In: Photonics East'99, International Society for Optics and Photonics, pp. 246–257 (1999)
4. Castano, A., Behar, A., Will, P.M.: The conro modules for reconfigurable robots. IEEE/ASME Trans. Mechatron **7**(4), 403–409 (2002)
5. Davey, J., Kwok, N., Yim, M.: Emulating self-reconfigurable robots-design of the smores system. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), pp. 4464–4469 (2012)
6. Eckenstein, N., Yim, M.: The x-face: an improved planar passive mechanical connector for modular self-reconfigurable robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012), pp. 3073–3078 (2012)
7. Eckenstein, N., Yim, M.: Area of acceptance for 3d self-aligning robotic connectors: concepts, metrics, and designs. In: proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2014) (2014 (in submission))
8. Fitch, R., Butler, Z.: Million module march: scalable locomotion for large self-reconfiguring robots. Int. J. Robot. Res. **27**(3–4), 331–343 (2008)
9. Fukuda, T., Nakagawa, S., Kawauchi, Y., Buss, M.: Structure decision method for self organising robots based on cell structures-cebot. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp 695–700 (1989)
10. Garcia, R.F.M., Hiller, J.D., Stoy, K., Lipson, H.: A vacuum-based bonding mechanism for modular robotics. IEEE Trans. Robot. **27**(5), 876–890 (2011)

11. Gilpin, K., Knaian, A., Rus, D.: Robot pebbles: one centimeter modules for programmable matter through self-disassembly. In: IEEE International Conference on Robotics and Automation (ICRA 2010), pp. 2485–2492 (2010)
12. Gilpin, K., Koyanagi, K., Rus, D.: Making self-disassembling objects with multiple components in the robot pebbles system. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3614–3621 (2011)
13. Girard, A.R., De Sousa, J.B., Hedrick, J.K.: Dynamic positioning concepts and strategies for the mobile offshore base. In: Proceedings of the IEEE International Conference on Intelligent Transportation Systems, pp. 1095–1101 (2001)
14. Gorbenko, A.A., Popov, V.Y.: Programming for modular reconfigurable robots. Program. Comput. Softw. **38**(1), 13–23 (2012)
15. Groß, R., Dorigo, M.: Self-assembly at the macroscopic scale. Proc. IEEE **96**(9), 1490–1508 (2008)
16. Kalontarov, M., Tolley, M.T., Lipson, H., Erickson, D.: Hydrodynamically driven docking of blocks for 3d fluidic assembly. Microfluid. Nanofluid. **9**(2–3), 551–558 (2010)
17. Karagozler, M.E., Kirby, B., Lee, W.J., Marinelli, E., Ng, T.C., Weller, M.P., Goldstein, S.C.: Ultralight modular robotic building blocks for the rapid deployment of planetary outposts (2006)
18. Klavins, E., Ghrist, R., Lipsky, D.: A grammatical approach to self-organizing robotic systems. IEEE Trans. Autom. Control **51**(6), 949–962 (2006)
19. Knaian, A.N., Cheung, K.C., Lobovsky, M.B., Oines, A.J., Schmidt-Neilsen, P., Gershenfeld, N.A.: The milli-motein: a self-folding chain of programmable matter with a one centimeter module pitch. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1447–1453 (2012)
20. Kotay, K., Rus, D., Vona, M., McGray, C.: The self-reconfiguring robotic molecule. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, pp. 424–431 (1998)
21. Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., Murata, S.: Distributed self-reconfiguration of m-tran iii modular robotic system. Int. J. Robot. Res. **27**(3–4), 373–386 (2008)
22. Murata, S., Kurokawa, H., Kokaji, S.: Self-assembling machine. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 441–448 (1994)
23. Murata, S., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S.: A 3-d self-reconfigurable structure. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, pp. 432–439 (1998)
24. Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., Kokaji, S.: Self-repairing mechanical systems. Autono. Robots **10**(1), 7–21 (2001)
25. Nilsson, M.: Heavy-duty connectors for self-reconfiguring robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002), vol. 4, pp. 4071–4076 (2002)
26. O'Hara, I., Paulos, J., Davey, J., Eckenstein, N., Doshi, N., Tosun, T., Greco, J., Seo, J., Turpin, M., Kumar, V., Yim, M.: Self-assembly of a swarm of autonomous boats into floating structures. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2014 (in submission))
27. Østergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the atron lattice-based self-reconfigurable robot. Auton. Robots **21**(2), 165–183 (2006)
28. Oung, R., Andrea, R.: The distributed flight array. Mechatronics **21**(6), 908–917 (2011)
29. Pamecha, A., Ebert-Uphoff, I., Chirikjian, G.S.: Useful metrics for modular robot motion planning. IEEE Trans. Robot. Autom. **13**(4), 531–545 (1997)
30. Park, M., Chitta, S., Teichman, A., Yim, M.: Automatic configuration recognition methods in modular robots. Int. J. Robot. Res. **27**(3–4), 403–421 (2008)
31. Piranda, B., Laurent, G.J., Bourgeois, J., Clévy, C., Möbes, S., Fort-Piat, N.L.: A new concept of planar self-reconfigurable modular robot for conveying microparts. Mechatronics **23**(7), 906–915 (2013)

32. Revzen, S., Bhoite, M., Macasieb, A., Yim, M.: Structure synthesis on-the-fly in a modular robot. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4797–4802 (2011)
33. Romanishin, J.W., Gilpin, K., Rus, D.: M-blocks: Momentum-driven, magnetic modular robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3073–3078 (2013)
34. Shen, W.M., Kovac, R., Rubenstein, M.: Singo: a single-end-operative and genderless connector for self-reconfiguration, self-assembly and self-healing. In: IEEE International Conference on Robotics and Automation (ICRA'9), pp. 4253–4258 (2009)
35. Shimizu, M., Ishiguro, A., Kawakatsu, T.: Slimebot: A modular robot that exploits emergent phenomena. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, pp. 2982–2987 (2005)
36. Sprowitz, A., Pouya, S., Bonardi, S., Van den Kieboom, J., Mockel, R., Billard, A., Dillenbourg, P., Ijspeert, A.J.: Roombots: reconfigurable robots for adaptive furniture. IEEE Comput. Intell. Mag. **5**(3), 20–32 (2010)
37. Støy, K.: An introduction to Self-Reconfigurable Robots. MIT Press, Boston, MA (2009)
38. Suh, J., Homans, S., Yim, M.: Telecubes: mechanical design of a module for self-reconfigurable robotics. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002), vol. 4, pp. 4095–4101 (2002)
39. Tolley, M.T., Krishnan, M., Erickson, D., Lipson, H.: Dynamically programmable fluidic assembly. Appl. Phys. Lett. **93**(25), 254,105–254,105–103 (2008)
40. Tolley, M.T., Kalontarov, M., Neubert, J., Erickson, D., Lipson, H.: Stochastic modular robotic systems: a study of fluidic assembly strategies. IEEE Trans. Robot. **26**(3), 518–530 (2010)
41. Unsal, C., Kiliccote, H., Khosla, P.K.: I (ces)-cubes: a modular self-reconfigurable bipartite robotic system. In: Photonics East'99, International Society for Optics and Photonics, pp. 258–269 (1999)
42. Vasilescu, I., Varshavskaya, P., Kotay, K., Rus, D.: Autonomous modular optical underwater robot (amour) design, prototype and feasibility study. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, pp. 1603–1609 (2005)
43. Vassilvitskii, S., Yim, M., Suh, J.: A complete, local and parallel reconfiguration algorithm for cube style modular robots. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA '02), vol. 1, pp. 117–122 (2002)
44. Wei, H., Chen, Y., Liu, M., Cai, Y., Wang, T.: Swarm robots: from self-assembly to locomotion. Comput. J. **54**(9), 1465–1474 (2011)
45. White, P., Kopanski, K., Lipson, H.: Stochastic self-reconfigurable cellular robotics. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA '04), vol. 3, pp. 2888–2893 (2004)
46. White, P.J., Yim, M.: Reliable external actuation for full reachability in robotic modular self-reconfiguration. Int. J. Robot. Res. **29**(5), 598–612 (2010)
47. White, P.J., Thorne, C.E., Yim, M.: Right angle tetrahedron chain externally-actuated testbed (ratchet): a shape-changing system. In: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE), vol. 7, pp. 807–817 (2009)
48. White, P.J., Latscha, S., Schlaefer, S., Yim, M.: Dielectric elastomer bender actuator applied to modular robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), pp. 408–413 (2011)
49. Yim, M., Zhang, Y., Duff, D.: Modular robots. IEEE Spectr. **39**(2), 30–34 (2002)
50. Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., Homans, S.: Modular reconfigurable robots in space applications. Auton. Robots **14**(2–3), 225–237 (2003)
51. Yim, M., Shirmohammadi, B., Sastra, J., Park, M., Dugan, M., Taylor, C.: Towards robotic self-reassembly after explosion. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2767–2772 (2007)
52. Yoshida, E., Matura, S., Kamimura, A., Tomita, K., Kurokawa, H., Kokaji, S.: A self-reconfigurable modular robot: reconfiguration planning and experiments. Int. J. Robot. Res. **21**(10–11), 903–915 (2002)

53. Yoshida, E., Murata, S., Kokaji, S., Kamimura, A., Tomita, K., Kurokawa, H.: Get back in shape![sma self-reconfigurable microrobots]. IEEE Robot. Autom. Mag. **9**(4), 54–60 (2002)
54. Zykov, V., Chan, A., Lipson, H.: Molecubes: an open-source modular robotics kit. In: Proceedings of the IROS, vol. 7 (2007)

Chapter 4
Lattice-Based Modular Self-Reconfigurable Systems

Kohji Tomita, Haruhisa Kurokawa, Eiichi Yoshida, Akiya Kamimura, Satoshi Murata and Shigeru Kokaji

Abstract This chapter presents a review of research related to self-reconfigurable systems at AIST, particularly addressing their lattice nature. Mainly, three systems are described: Fractum in a 2D hexagonal lattice, and 3D units and M-TRAN in a cubic lattice. Each has distinctive characteristics. Their basic design, reconfiguration methods, and physical implementation issues are discussed and compared.

4.1 Introduction

Typical conventional robots and mechanical systems comprise components of various kinds such as structural elements, mechanisms, and actuators. Components are arranged carefully. The mechanical connectivity among the components does not change. When a robot's task is fixed or its environment is known well in advance, such a design is crucial for realizing high precision, durability, and efficiency in terms of time, space, and energy.

K. Tomita (✉) · H. Kurokawa · E. Yoshida · A. Kamimura
National Institute of Advanced Industrial Science and Technology (AIST),
Tsukuba, Japan
e-mail: k.tomita@aist.go.jp

H. Kurokawa
e-mail: kurokawa-h@aist.go.jp

E. Yoshida
e-mail: e.yoshida@aist.go.jp

A. Kamimura
e-mail: kamimura.a@aist.go.jp

S. Murata
Department of Bioengineering and Robotics, Tohoku University, Sendai, Japan
e-mail: murata@molbot.mech.tohoku.ac.jp

S. Kokaji
Ibaraki, Japan
e-mail: s_kokaji@mail1.accsnet.ne.jp

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_4

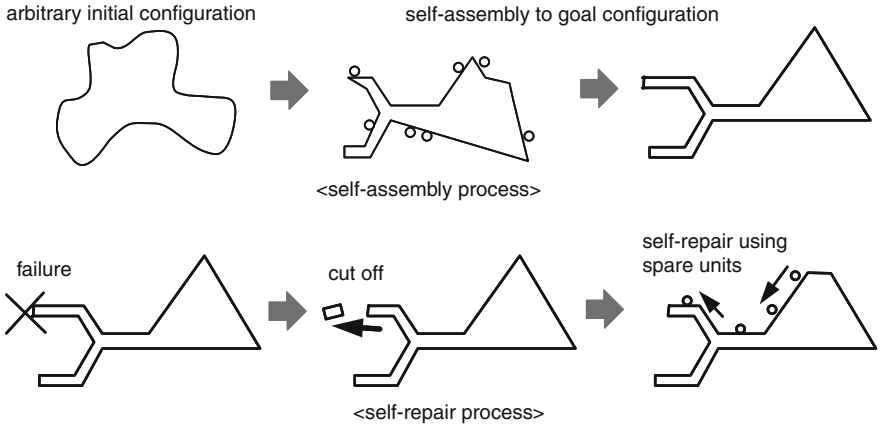


Fig. 4.1 Self-assembly and self-repair [1]

We have been working on robotic systems comprising many identical components called *units*, similar to living systems with their constituent cells and identical genetic information. The units are assumed to have the ability to change their mutual connections independently. Such systems, called modular self-reconfigurable systems, have the following potential benefits.

- Adaptability: Depending on the change of its environment or task, the system can adapt its configuration and function.
- Scalability: Depending on the number of units, different structure and function can be realized.
- Redundancy: Malfunctioning units can be replaced.
- Reliability: The system can continue working in spite of component failures.

Modular self-reconfigurable systems are well suited for self-assembly and self-repair, which is difficult to realize using conventional robots. A process of self-assembly and self-repair is presented in Fig. 4.1. The system assembles itself into a target configuration and achieves its function. If some part is damaged, the system repairs itself and recovers its functionality by cutting off the damaged parts and reassembling itself again using undamaged spare units.

For designing modular self-reconfigurable systems, it is helpful to make a lattice constraint so that the rest positions of the units are restricted to the grid points of a lattice. The translational and rotational symmetry and the regularity of lattices reduce the complexity of modular self-reconfigurable systems both computationally and mechanically. Computationally, it is important that each unit has at most a certain fixed number of neighboring units. For instance, by assigning a discrete state to each unit and by providing state transition rules described in a uniform manner, the global state or configuration of the system can be controlled as in cellular automata [2, 3]. Mechanically, restricting the rest positions of the units to grid points reduces complexity in the design of motions of the units. For instance, global reconfiguration

can be achieved by repetitive local reconfiguration such that a unit on a grid point moves to an adjacent vacant grid point. Such local reconfiguration is much simpler because the possible movements are few.

In spite of the theoretical simplicity, their hardware implementation brings much difficulty. Numerous lattice-based modular self-reconfigurable robots have been proposed [1, 4–6]. Many have only been implemented in a few hardware units or only in simulation. This chapter presents a review of our three lattice-based hardware systems: Fractum, 3D units, and M-TRAN. Among these three systems, the former two were intended mainly to simulate an ideal lattice-based system. M-TRAN was intended to function by itself as a real machine or a robot capable of robotic motion by continuous actuation. This survey presents a review of their basic designs, reconfiguration algorithms, and hardware implementation, and discussion of general problems of lattice-based mechanical systems.

4.2 Fractum

The Fractum is our early prototype system based on the 2D hexagonal lattice [7–9]. Using this system, we developed several algorithms for self-assembly and self-repair. Then we conducted experiments to demonstrate its feasibility.

4.2.1 Basic Design

Figure 4.2 is a schematic view of a Fractum unit. No movable part exists in any single unit. Self-reconfiguration is done by controlling magnetic force.

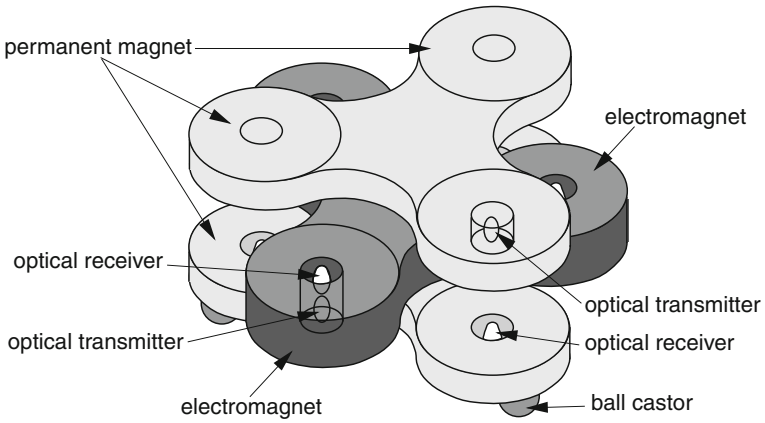


Fig. 4.2 Schematic view of Fractum [1]

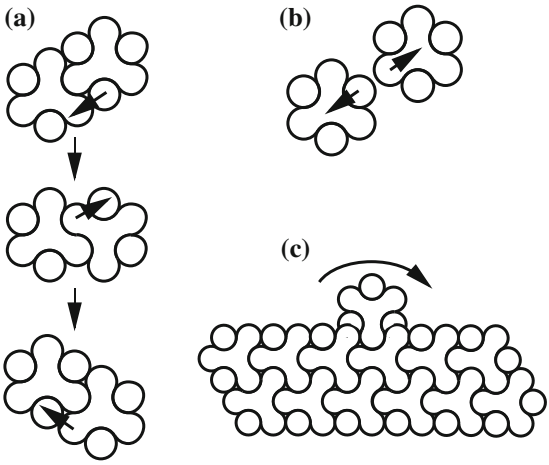


Fig. 4.3 Basic functions of Fractum. **a** Change connection, **b** Cut connection, **c** Transportation [1]

A Fractum unit has a three-layer structure. Permanent magnets are placed in the round parts of the top and bottom layers. Electromagnets are placed in the round parts of the middle layer. The round parts are called arms. By attractive or repulsive force between an electromagnet of a unit and permanent magnets of another unit, the arm of the former unit is attracted to or repelled from the arm of the latter unit. Using this, as shown in Fig. 4.3, changing the polarity of electromagnets realizes (a) change of connective relation between neighbor units, (b) connection cut, and (c) transportation of a unit.

When two units are connected, bidirectional communication is possible between them using the optical transmitters and receivers embedded in the arms. It is used for transmitting units' states.

4.2.2 Algorithm I

A reconfiguration method based on the local connective relation is presented. To make the best use of physically distributed characteristics of the units, it is desirable that every unit have the same software and that the overall system be controlled in a distributed manner without global information to the greatest extent possible. We consider describing a global configuration as a collection of local connective relations.

Each Fractum unit has six connecting arms, and their connective relations are classified into the 12 types shown in Fig. 4.4a. Using these types, the global configuration is described. For instance, a triangle in Fig. 4.4b includes three types: o, K and s. Every type o unit has two neighbors with type K. Such a local connective relation is written as 'o (K, K)'. Similarly, units with type K have neighbors with type o, K, K, and s, and units with type s have six neighbors with type s. This configuration is

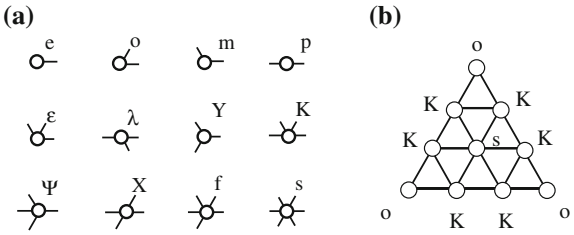


Fig. 4.4 Connection types and triangular configuration [1]

therefore described as

o(K, K),
K(o, K, K, s),
s(K, K, K, K, K, K).

We regard it as a target description.

To achieve a target configuration from an arbitrary initial configuration, each unit is assumed to be given a common target description. Then, depending on the local connective information obtained by communication with the neighbors, each unit repeatedly changes its connection if necessary. The outline of the process is as follows. Each unit calculates its distance of the current local configuration to the given target configuration. If this distance is zero for a unit, then the current local configuration matches some part of target configuration and the unit need not move. If the distance is zero for all units, then the process is regarded as completed and no unit moves. Otherwise, a unit with large distance moves randomly to the right or left.

Hardware experiment of this method using 11 units with the target description (1) is shown in Fig. 4.5.

4.2.3 Algorithm II

Using the method described above, when the structure is large, the system does not always converge to the goal configuration. To improve the success rate and speed of

Fig. 4.5 Reconfiguration experiments using Fractum [1]

convergence, it is effective to provide information not only of the final configuration, but of the intermediate configuration in each construction step.

Figure 4.6 presents an image of the second algorithm. The idea is that a connection network is developed hierarchically from a unit called a kernel, and that undifferentiated units, which are encircling the circumference, are supplied at necessary locations.

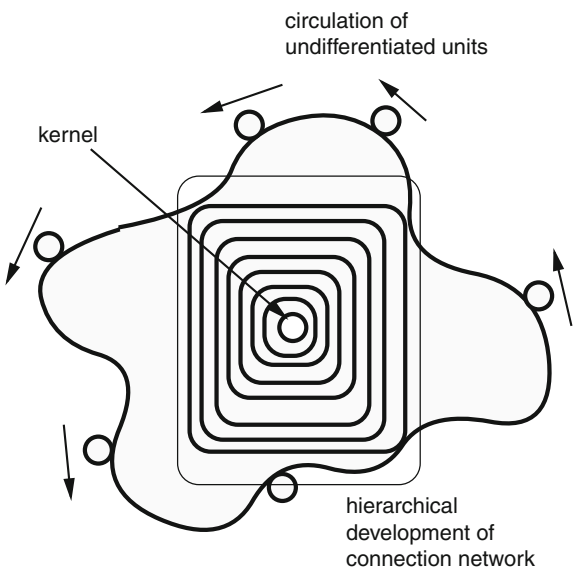
Information at each stage is described using the connection types explained above in the form of a lower triangular matrix called a description matrix. From a given target shape, its corresponding description matrix can be generated automatically. Figure 4.7 shows an example of a description matrix and its corresponding configuration at each step. The connection network is generated step-by-step, and the connection type of each unit at each step is described in the matrix. For instance, at stage 4, three units are added, all with type o. They change their types to ε at stage 5, and have type K at stage 6 and later.

When a failure occurs, the system can execute self-repair by cutting off the faulty part and returning to a previous stage. Figure 4.8 shows a simulation of self-assembly and self-repair using this method.

4.2.4 Meta Unit

A group of several units that are arranged in a larger lattice and which have reconfiguration capability as a group in the larger lattice are called meta units. Meta units are intended not only to mimic the original behavior, but also to have more flexibility.

Fig. 4.6 Algorithm II [1]



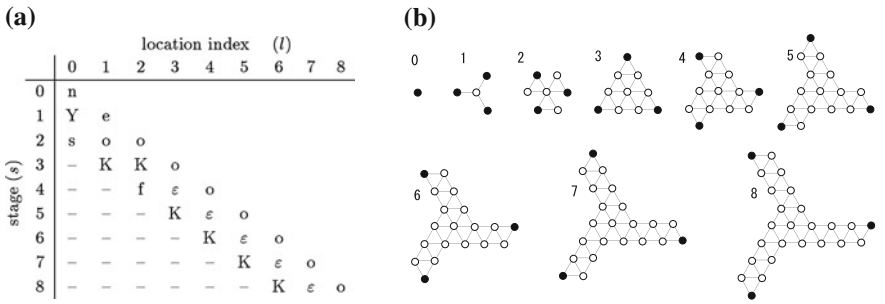


Fig. 4.7 Description matrix and configuration at each stage [1]

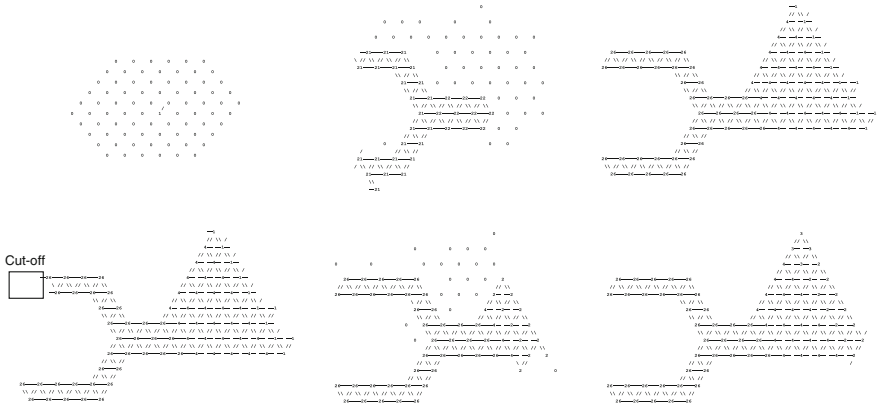


Fig. 4.8 Simulation of self-assembly by Algorithm II

For instance, one can consider a meta unit for Fractum. Using the original units, in the configuration in Fig. 4.9a, it is not possible for the gray unit to move downward and pass through a gap. Using a meta unit composed of 24 original units in Fig. 4.9b, however, by appropriately designing a moving sequence, the gray meta unit can pass through a gap and perform the prohibited motion (Fig. 4.9c).

From a practical perspective, implementing meta units requires many original units and is not so easy. However, when the movability of the original unit is not enough, introducing meta units is effective, as described later with M-TRAN.

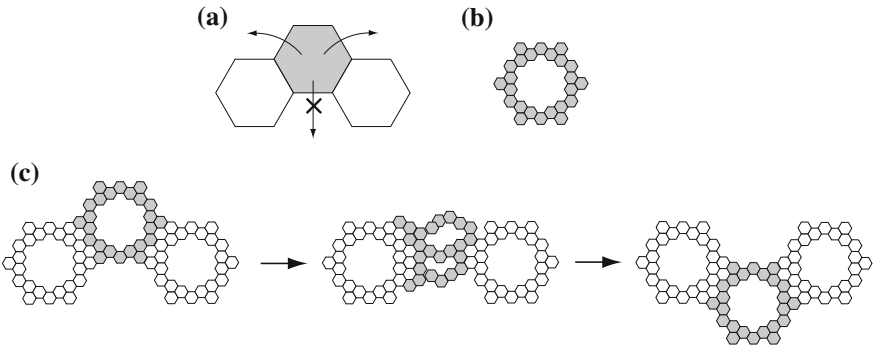


Fig. 4.9 Meta unit for Fractum [1]

4.3 3D Units

Our next system is Three Dimensional Universal Connection System (called as 3D units) [10], which is an extension of 2D hexagonal lattice of Fractum to cubic lattice.

4.3.1 Design

Each 3D unit occupies a grid point in the cubic lattice. Figure 4.10a shows a schematic view of a unit. Each unit has one rotation arm with a connecting hand in every six directions. The hand is for connection and disconnection. The arm is for local reconfiguration by pairwise motion; rotation of itself and its neighbor depending on connective situation. Such rotation enables local reconfiguration as shown in Fig. 4.10b, c. We assume that unit X and unit Y are connected, and that both are connected to their own lower units. Reconfiguration is performed by the following steps:

1. Unit Y and unit Z cut their mutual connection.
2. Unit X rotates the lower arm for 90° around the b axis, resulting in the position change of unit Y.
3. Unit Y and its lower unit connect each other.

Figure 4.11 shows the developed hardware units.

4.3.2 Reconfiguration

A reconfiguration method based on local connection types, which is an extension of the Algorithm I for Fractum, was developed.

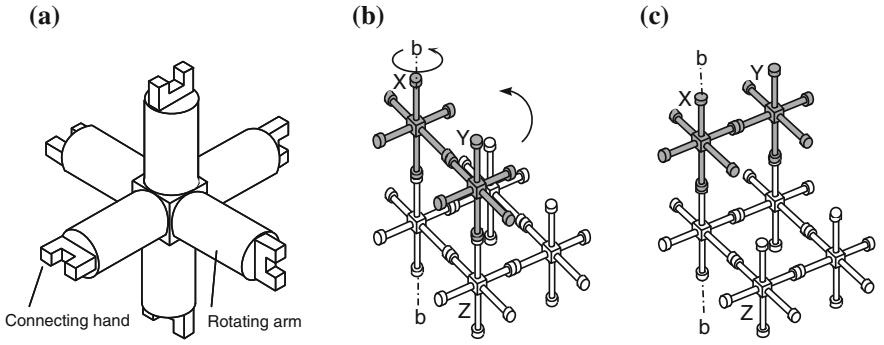
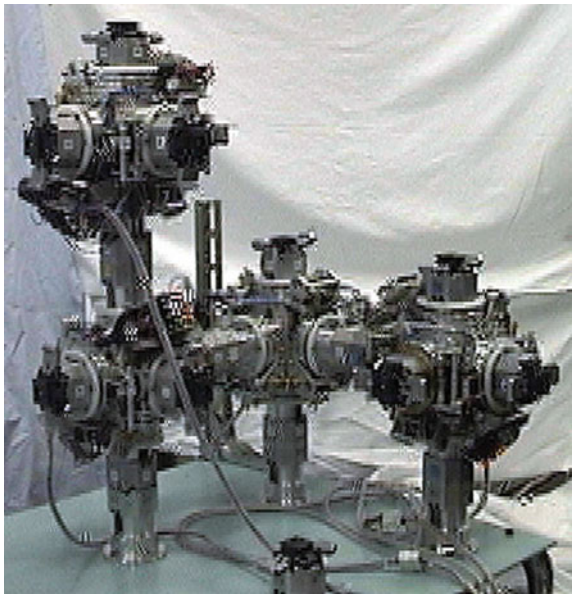


Fig. 4.10 Schematic view of the 3D unit and basic function

Fig. 4.11 3D units hardware



In this system, nine connection types exist, as shown in Fig. 4.12a. Figure 4.12b shows simulation steps using 12 units for constructing a target shape described as

C31(C31, C31, C41),
C41(C31, C31, C41, C41).

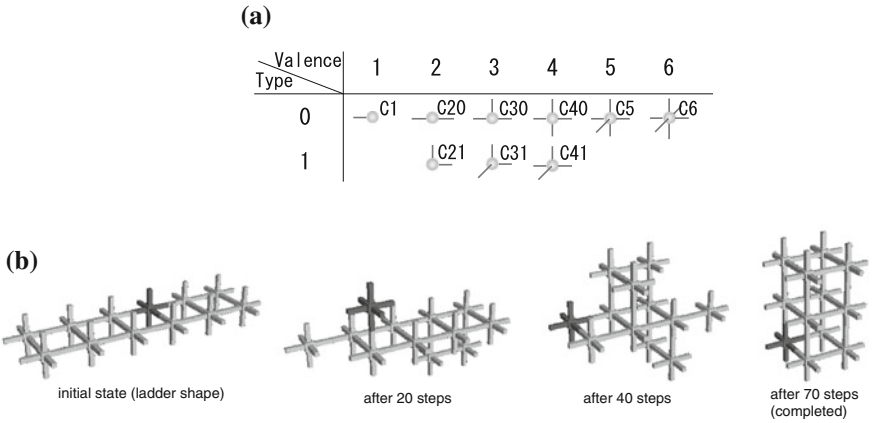


Fig. 4.12 Connection types and reconfiguration simulation of 3D units. **a** Connection types of 3D units, **b** Reconfiguration simulation

4.4 M-TRAN

We developed Modular TRANsformer (M-TRAN) by simplifying the 3D units so that two units are mechanically combined into a module [11–17]. Each module has only two rotational degrees of freedom, but modules have 3D reconfiguration capability as a group.

4.4.1 Design

An M-TRAN module comprises three parts: an active block, a passive block, and a link between them, as in Fig. 4.13. Each block has the shape of a half-cube and half-cylinder, and can rotate 180°. Each block has three flat surfaces for connection. An active block connects only with a passive block of another module. Because of the parity property of the cubic lattice, this polarity is not a restriction.

In contrast to the previous two systems, M-TRAN is largely asymmetric, i.e., possible motions by an M-TRAN module differ depending on its posture. For example, if we assume that a single module is placed on a plane filled with modules as in Fig. 4.14, the module can move by rolling (a) or pivoting (b), depending on its posture. These two postures can be changed mutually with the help of an additional module as in (c). In many reconfiguration sequences of M-TRAN, cooperation of two or more modules in this manner is indispensable.

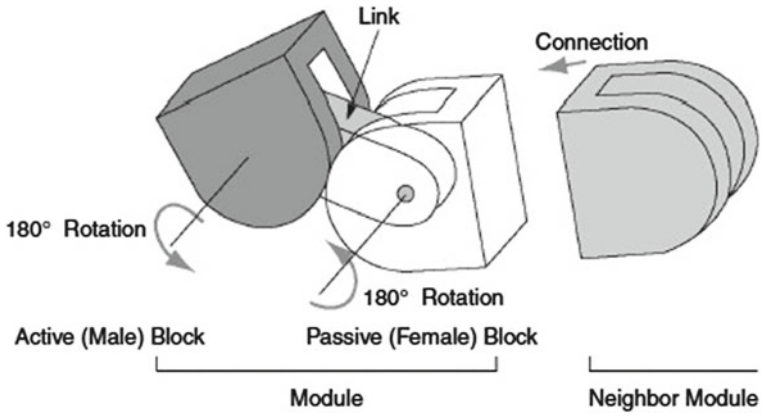


Fig. 4.13 Schematic view of M-TRAN module

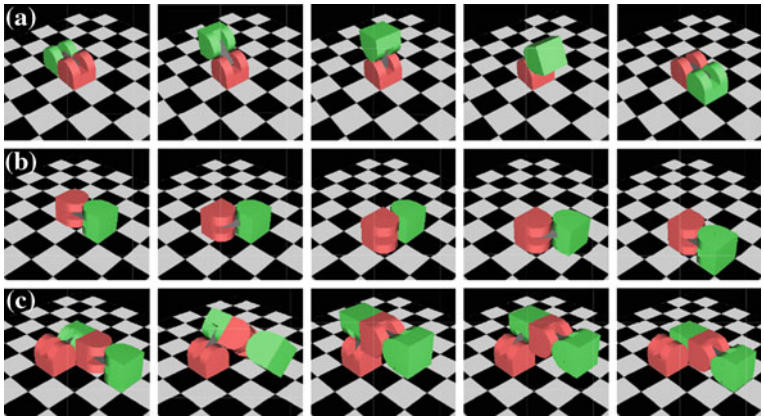


Fig. 4.14 Motions of M-TRAN modules

4.4.2 Reconfiguration

Designing the reconfiguration procedure of M-TRAN modules is not straightforward because the possible motion and connection of a module is restricted by its simplicity and anisotropy. Two neighboring modules in the lattice cannot always be mutually connected because of the limited number of connecting surfaces on each block. A module's motion is also restricted by nearby modules because of collision.

We have manually developed various reconfiguration sequences for small scale reconfiguration up to 10 modules, including transformation of a four-legged structure to a linear structure and its reversal (Fig. 4.15a, e).

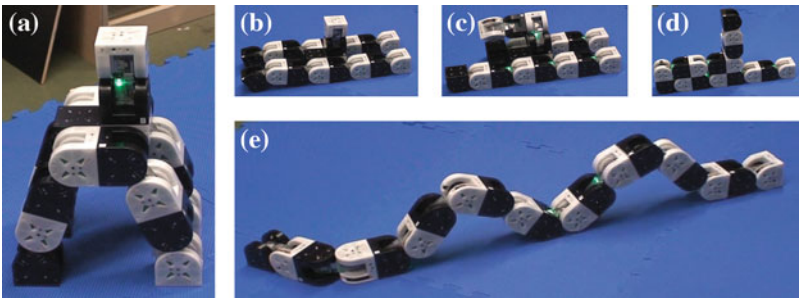


Fig. 4.15 Reconfiguration experiments from a four-legged robot to a linear robot [1]

This small scale reconfiguration is from one initial configuration to another. Therefore, it is different from self-assembly as discussed in the previous sections. To realize self-assembly for a larger reconfiguration, using meta units and introducing regularity to the whole system is beneficial. Herein, we present some regular structures for reconfiguration.

The first is shown in Fig. 4.16a: meta-modules, each of which comprises four modules, are connected linearly. A pair of modules called a converter is attached. This structure is capable of flow motion by the repetitive reconfiguration of transporting the tail meta module to the head position. Turning horizontally or vertically is also possible when assisted by the converter.

The second one (Fig. 4.16b) is a simpler structure for linear flow motion. It comprises two lines of modules. Reconfiguration of this structure was confirmed experimentally.

The next structure is two-dimensional, as shown in Fig. 4.16c. Each meta module comprises four modules. If this structure with sufficient number of modules is placed on a plane, then it can move to a desired direction by flow motion and can take various 2D shapes (Fig. 4.16d).

The final structure is three-dimensional (Fig. 4.17). Each meta module is the same as the previous one, but meta modules are connected in a different way to constitute the cubic lattice. Local reconfiguration procedures for meta modules were developed, but it will be difficult to realize on earth because of gravity.

Individual reconfiguration processes for each step motion require much communication and control including connection and disconnection and rotation, but discussion of the related details is omitted here. By describing them as state transition, the processes can be, in principle, formulated directly as finite state automata.

4.4.3 Robotic Motion

We briefly introduce robotic motion by M-TRAN modules by relaxing the lattice constraint. After reconfiguration into an appropriate structure under a lattice constraint,

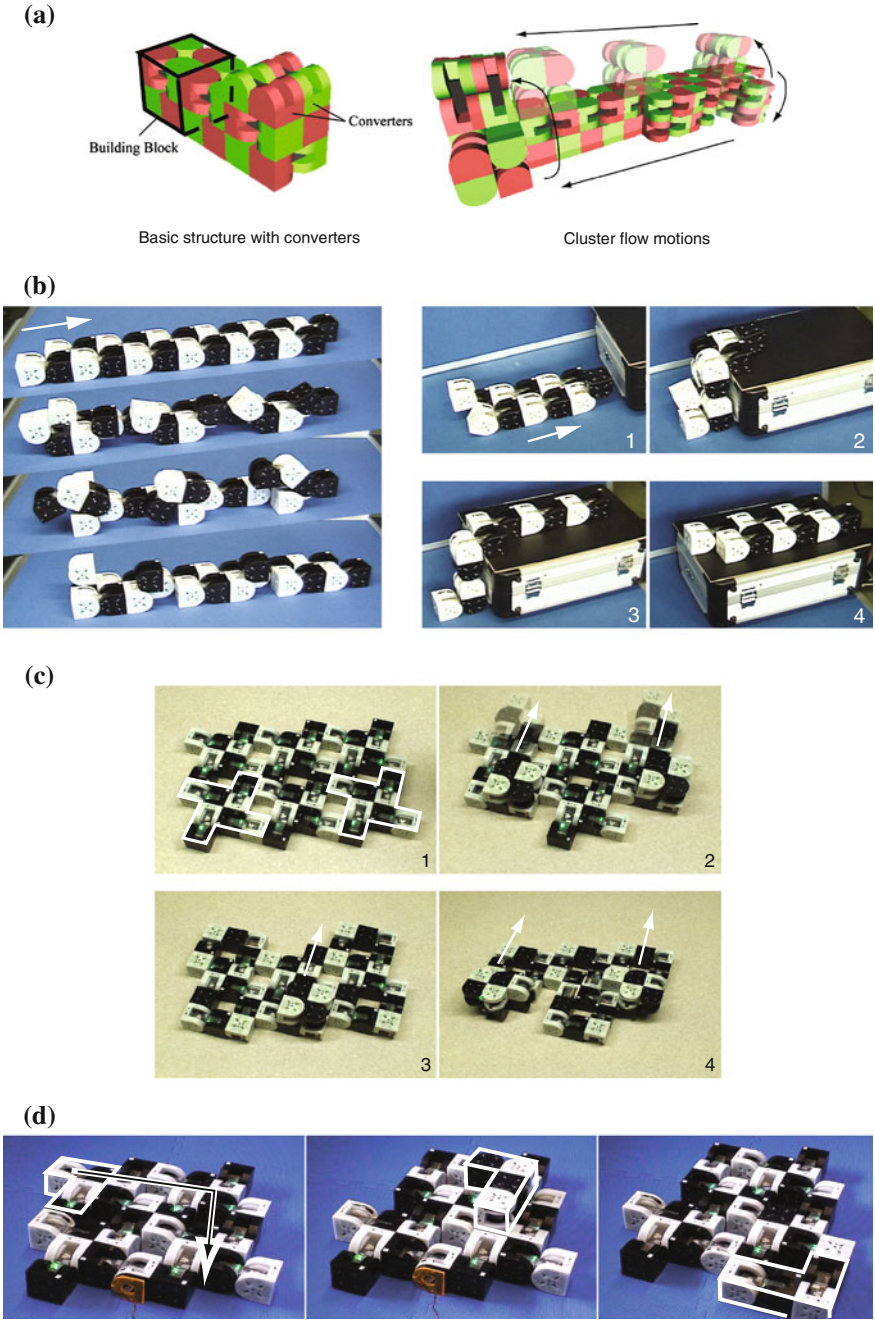


Fig. 4.16 Regular structures of M-TRAN [1]

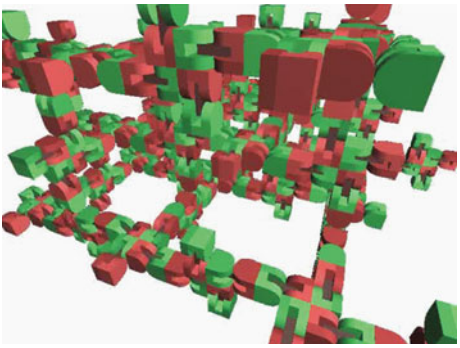


Fig. 4.17 3D regular structure of M-TRAN

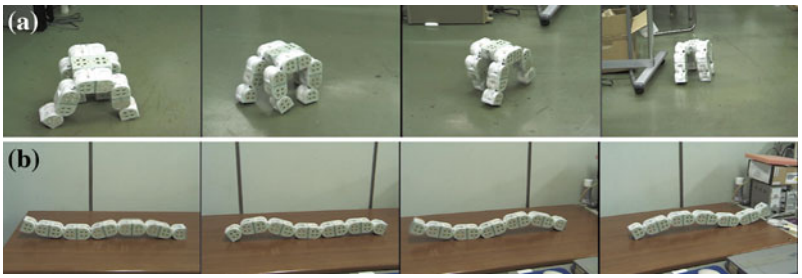


Fig. 4.18 Various locomotion patterns

the modules as a whole can perform robotic motions such as walking. Figure 4.18 shows locomotion experiments of four-legged and linear robots. In fact, locomotion control for each robot is based on a central pattern generator (CPG), the network of which was obtained by a genetic algorithm.

4.5 General Problems of Lattice-Based Mechanical Systems

In this section, we examine hardware issues of general lattice-based systems. Among the three systems described above, the former two, namely Fractum and the 3D units, were intended mainly to simulate ideal lattice-based systems. An ideal system is a distributed physical system comprising many identical units. Each unit is considered simple both in motion and in information processing capabilities and each produces motion according to local rules or local decision making such as a cellular automaton. It can be as small as a molecule or a biological cell, so physical and mechanical realization was not the main objective, and mechanical performance was not an issue. Development of these systems actually left behind many physical problems.

The three generations of M-TRAN, namely M-TRANs I, II, and III, were developed to overcome such problems in mechanical realization. It was aimed to work by itself as a real machine or a robot. Designs of two types were combined, which were designated afterwards as lattice-based and chain-based. Improved performance of the unit was necessary because a chain-based modular robot as a whole must work as a locomotive robot (Fig. 4.18) under rather centralized control.

4.5.1 Improvement in M-TRAN Hardware

An M-TRAN module equips stronger actuators and a more powerful microprocessor than the other two systems. Each cubic block of an M-TRAN module equips only one actuator for motion, so a module needs to carry other modules for reconfiguration. For robotic locomotion, such as walking in a legged configuration (Fig. 4.18), a small number of joint motors must support and move the whole body, so a stronger actuator is necessary. At the same time, joints' motion must be synchronized among modules, and modules need to respond fast to an input to a module's sensor. If dynamic control is sought, more complicated numerical processing is necessary. Therefore, an M-TRAN III module has a fast processor (32 bit CPU, SH-II; Renesas Electronics Corp.) and fast and global inter-module communication (CAN bus).

Each module of M-TRAN II and III has a battery. This not only made the whole body locomotion possible but also helped reconfiguration as a lattice-based system. For Fractum and 3D units, their reconfiguration invariably made tethers for power supply tangle. To address this problem, M-TRAN I was designed so that a single set of tethers is connected at one end of the whole system and power is transmitted to all the units via connection. This method, however, proved to be ineffective because the transmission loss was too large when several units were connected serially.

4.5.2 General, Physical Problem in Modular Reconfigurable Systems

As a lattice-based system, M-TRAN is based on a cubic lattice, but a single M-TRAN module comprises two cubic units. This design presents the important benefits of solving problems encountered by Fractum and the 3D units. Simultaneously, it presents many shortcomings. Each such problem and corresponding design solution relates to others. It is not easy to analyze problems systematically and to derive a design guide. Here, some general and important issues will be explained along with a brief introduction of others [1].

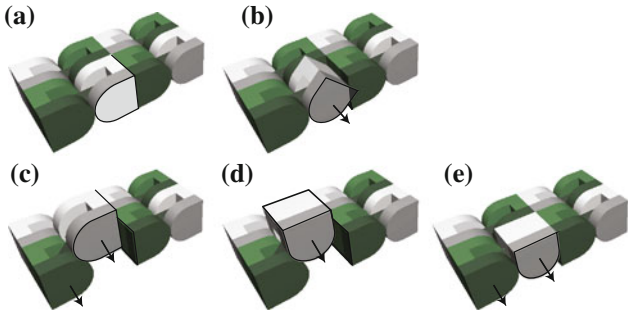


Fig. 4.19 Collision avoidance by parallel axis motion. Although collision is unavoidable by a single axis motion as in **b**, two axis motion avoids it as **c**, **d**, and **e**

4.5.2.1 Unit Shape

With a lattice-based logical system, it is considered that any unit in a cell can move to its adjacent neighbor void cell based on its state, its neighbors' states, and the rules. With a physical system, such motion is not always possible depending on the unit shape, kinematic design, local configuration, and physical performance of actuators. Although a circular or spherical unit can move without colliding with other units, a square or cubic unit cannot always do so, depending on its motion mechanism and surrounding units. However, a circle or a sphere contacts with its neighbor only via a point, which makes it difficult to ensure a rigid connection and strong actuation. A square or a cube is beneficial not only in terms of precision, rigidity of connection, and in actuation strength. It also makes a larger interior space useful for the installation of mechanisms, actuators and circuits. In the case of M-TRAN, with its semi-cubic shape, a problem of collision is partially relaxed by two parallel axes of a single module (Fig. 4.19).

4.5.2.2 Geometric Error, Structure, and Deformation

In contrast to identical units of an ideal lattice-based system, any mechanical product invariably has errors in its geometry. In actual production performed by a precision machine, elements are selected and combined with others so that the error of one element compensates an error of others. Regarding a modular system, all the units must fit at any position. Such compensation is not possible (Fig. 4.20).

With geometric errors, multiple units cannot fit perfectly, so the elasticity of units is necessary to assemble units. In the usual case of assembling a machine or a structure, no element is assembled tightly when it is assembled to others, but all the elements are tightened gradually to distribute stress and strain over the whole body. Such gradual tightening is not considered for a lattice-based machine. Therefore, when units are

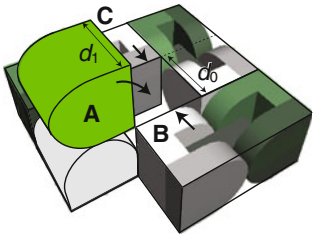


Fig. 4.20 Problem of geometric error. If $d_1 > d_0$, then A cannot move into the gap separating B and C

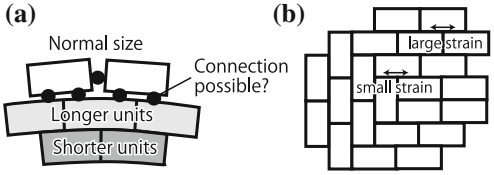


Fig. 4.21 Building a large structure. **a** Irregular units, **b** Non-uniform strain

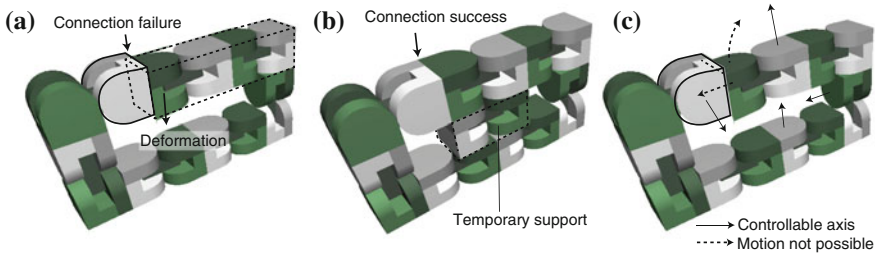


Fig. 4.22 Deformation and control. **a** Gravitational bending of a cantilever beam might cause large misalignment and connection failure. **b** Deformation in **a** is avoided using a supporting module. **c** Without sufficient motion DOFs, deformation in **a** cannot be corrected by control

assembled adding one after another, error will accumulate so that with some number of units, a new unit cannot be added without violating some deformation limit or an actuator's maximum output (Fig. 4.21).

However, elasticity cannot be set too weak. With weak elasticity, deformation of a cantilever beam under gravity might become so large that other units cannot move into the space under the beam or large positioning error might produce an unsuccessful connection (Fig. 4.22a).

Consequently, elasticity and rigidity are required simultaneously. Their proper balance cannot generally be attained using a conventional mechanical system. Regarding M-TRAN, this problem is sometimes avoided by the meticulous design of a reconfiguration sequence. For example, such a vulnerable part as a cantilever beam is

temporarily supported by another module (Fig. 4.22b). This method of avoidance, however, tends to require planning of the whole system. For that reason, a distributed and decentralized nature of a lattice-based system is not always attained.

4.5.2.3 Density

Because an M-TRAN module lacks full symmetry for a cubic unit, it must carry at least one other unit for reconfiguration; its actuator needs to lift its half part plus at least another full module. When an actuator and a battery of sufficient power are given, the weight of a module is roughly determined. With a given weight, and with available force and torque, the smaller the module, the greater the number of modules that can be lifted. Therefore, a higher density of a module is better. Actually, M-TRAN II and III are, although not optimized, quite packed. With its current size, i.e., of about 650 mm cube, it's unlikely that drastic, mechanical improvement in M-TRAN can be made if it is produced using currently available technology.

4.5.2.4 Control

Robotics can be regarded as a science of intelligence in mechanical control. If a module's geometry is not sufficiently precise or if structural deformation is not negligible, then some control or adaptation should be integrated into the system.

Designing a module to have controllability of geometry might make the problem worse, making the module more complicated and heavy, hence more vulnerable to external forces and easier to deform. For a single module, to measure or correct misalignment as in Fig. 4.22a is not easy and mostly impossible. Cooperation of multiple modules is also not a solution, because sensors and actuators of surrounding modules do not always possess sufficient degrees of freedom for feedback control as shown in Fig. 4.22c. Consequently, a straightforward approach to feedback control is not practical, and is, from the very first, not a proper approach to a lattice-based mechanical system, in which an ideal unit is presumably able to move from one cell to another with a switching motion.

For M-TRAN, each actuator is position-controlled mostly at five discrete angles in 45 deg step. There is no position sensor except angle sensors for the servo controllers and a sensor detecting completion of connection with another module. With such a setting, the modules' motions were controlled in a feed-forward manner. Feasible sequences of reconfiguration such as those in Figs. 4.15 and 4.16c were designed and examined in experiments using a trial-and-error process.

4.5.3 Achievement by M-TRAN

In case of small-scale self-reconfiguration, in which up to 10 modules change their configurations among pre-designed robotic ones, M-TRAN III modules realized

many steps of lattice-based transformations with a small failure ratio. In such cases, geometric errors were not severe. Because robotic motions are intended after reconfiguration, synchronized re-tightening of the whole connection was made to relax possible concentrated strain in the whole structure, thereby avoiding the problems described above in Sect. 4.5.2.2.

Self-reconfiguration in a larger structure as shown in Fig. 4.16c, avoided the problems presented in Fig. 4.20 and Fig. 4.21 because the structure is planar and is not fully packed but instead contains spaces. Some trials of reconfiguration experienced connection failures because of the problems explained in Sects. 4.5.2.2 and 4.5.2.4. So various sequences of reconfiguration for the same target configurations were designed and experimented, and better ones were selected.

Truly three-dimensional structures resembling those shown in Fig. 4.16d and their self-reconfiguration were not tried. They seem far from feasible by the current M-TRAN system because of the problems and difficulties described above.

4.6 Conclusions

We have reviewed self-reconfigurable robots particularly addressing their lattice nature. Three systems, Fractum, 3D units, and M-TRAN, were described. Hardware problems were discussed while taking M-TRAN as a typical case.

Although all the systems, based on latest mechanical and electronic technology, have achieved fairly useful results verifying a lattice-based system, assigning too much emphasis to such mechanical and electronic realization might lead to a dead end. Considering discussions presented in the previous section, the feasibility of a lattice-based mechanical system in general is not assured even when several numbers of units are developed and only with them, varieties of lattice-based motions are verified by experimentation. A breakthrough beyond the achievements of research and evaluations of modular robotics conducted in the past will require considerable numbers of new ideas related to design and a new method of manufacturing and working with vast amounts of units in a simple manner and at low cost.

Recent technologies, such as microfabrication and nanofabrication, DNA tiling, and other molecular nanotechnologies, seem promising. For example, DNA nanotechnology can produce molecular-scale components such as structures, logic gates, sensors, and actuators, which, if integrated properly, can realize molecular scale robots. Working in liquid might mitigate most of the problems listed above. Moreover, a chemical power supply can be made to all units without tethers and batteries. Controllable microparticles or microshells are anticipated for various applications such as drug delivery, such that drugs are contained, transported, and released by micro shells.

For the future development of such systems, the study of robotics is no doubt indispensable. Moreover, the fusion of diverse disciplines of science and technologies will be strongly required, such as nanotechnologies, molecular biology, information technology, and systems science including the study of lattice-based systems.

References

1. Murata, S., Kurokawa, H.: Self-Organizing Robots. Springer, Berlin (2012)
2. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. Int. J. Robot. Res. **23**(9), 919–937 (2004)
3. Stoy, K.: Using cellular automata and gradients to control self-reconfiguration. Robot. Autonom. Syst. **54**(2), 135–141 (2006)
4. Murata, S., Kurokawa, H.: Self-reconfigurable robots. IEEE Robot. Autom. Mag. **14**(1), 71–78 (2007)
5. Stoy, K., Brandt, D., Christensen, D.J.: Self-Reconfigurable Robots: An Introduction. MIT Press, Cambridge (2010)
6. 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 Robot. Autom. Mag. **14**(1), 43–52 (2007)
7. Murata, S., Kurokawa, H., Kokaji, S.: Self-assembling machine. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 441–448 (1994)
8. Tomita, K., Murata, S., Kurokawa, H., Yoshida, E., Kokaji, S.: Self-assembly and self-repair method for a distributed mechanical system. IEEE Trans. Robot. Autom. **15**(6), 1035–1045 (1999)
9. Yoshida, E., Murata, S., Tomita, K., Kurokawa, H., Kokaji, S.: An experimental study on a self-repairing modular machine. Robot. Autonom. Syst. **29**(1), 79–89 (1999)
10. Murata, S., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S.: A 3-D self-reconfigurable structure. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 432–439 (1998)
11. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. IEEE/ASME Trans. Mechatron. **10**(3), 314–325 (2005)
12. Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., Murata, S.: Distributed self-reconfiguration of M-TRAN III modular robotic system. Int. J. Robot. Res. **27**(3–4), 373–386 (2008)
13. Kurokawa, H., Yoshida, E., Tomita, K., Kamimura, A., Murata, S., Kokaji, S.: Self-reconfigurable M-TRAN structures and walker generation. Robot. Auton. Syst. **54**(2), 142–149 (2006)
14. Murata, S., Tomita, K., Yoshida, E., Kurokawa, H., Kokaji, S.: Self-reconfigurable robot. In: Procedings of International Conference on Intelligent Autonomous Systems, pp. 911–917 (1999)
15. Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., Kokaji, S.: M-TRAN: self-reconfigurable modular robotic system. IEEE/ASME Trans. Mechatron. **7**(4), 431–441 (2002)
16. Ostergaard, E.H., Tomita, K., Kurokawa, H.: Distributed metamorphosis of regular M-TRAN structures. In: Distributed Autonomous Robotic Systems 6, pp. 169–178. Springer (2007)
17. Yoshida, E., Matura, S., Kamimura, A., Tomita, K., Kurokawa, H., Kokaji, S.: A self-reconfigurable modular robot: reconfiguration planning and experiments. Int. J. Robot. Res. **21**(10–11), 903–915 (2002)

Chapter 5
Speed Control on a Hexapodal Robot Driven by a CNN-CPG Structure

E. Arena, P. Arena and L. Patané

Abstract Locomotion control in legged robots is an interesting research field that can take inspiration from biology to design innovative bio-inspired control systems. Central Pattern Generators (CPGs) are well known neural structures devoted to generate activation signals to allow a coordinated movement in living beings. Looking in particular in the insect world, and taking as a source of inspiration the *Drosophila melanogaster*, a hierarchical architecture mainly based on the paradigm of a Cellular non-linear Network (CNN) has been developed and applied to control locomotion in a fruit fly-inspired simulated hexapod robot. The modeled neural structure is able to show different locomotion gaits depending on the phase locking among the neurons responsible for the motor activities at the level of the leg joints and theoretical considerations about the generated pattern stability are discussed. Moreover the phase synchronization between the leg, altering the locomotion, can be used to modify the speed of the robot that can be controlled to follow a reference speed signal. To find the suitable transitions among patterns of coordinated movements, a reward-based learning process has been considered. Simulation results obtained in a dynamical environment using a *Drosophila*-inspired hexapod robot are here reported analyzing the performance of the system.

5.1 Introduction

Gait generation and locomotion control in artificial systems are extremely important to build efficient and highly adaptive walking robots. Since the last decade a huge effort has been paid to discover and model the rules that biological neural systems

E. Arena · P. Arena (✉) · L. Patané
Department of Electrical, Electronic and Computer Science Engineering, University of Catania, I-95125 Catania, Italy
e-mail: parena@dieei.unict.it

L. Patané
e-mail: lpatane@dieei.unict.it

P. Arena
National Institute of Biostructures and Biosystems (INBB), Viale delle Medaglie d'Oro 305, 00136 Rome, Italy

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_5

adopt to show efficient strategies for generating and controlling the gait in animals and manage the efficient transition among different patterns of locomotion. The work here presented is in line with the on-going studies on the insect brain architecture [1, 2]. In particular a huge effort has been paid recently to design block-size models for a number of different parts of the fly *Drosophila melanogaster* brain, to try to attain perceptual capabilities and to transfer them to biorobots. In the field of Bio inspired cognitive Robotics, the paradigm of Cellular Nonlinear Networks, the continuous time extension of cellular automata, has been widely exploited for their capabilities of complex spatial temporal pattern formation, both in the steady state regime [3, 4], and through dynamical attractors [5].

Regarding the Neurobiological studies on the fly motor control, while it is already known which are the centers involved in visually guided orientation control behaviors (i.e. Central Complex) [6–8], it is not clear how the high level controller acts at the low level, to finely modulate the neural circuitry responsible for the locomotion pattern generation, steering activities and others. On the other side, behavioral experiments are in line with the idea that the fruit fly mainly adopts the Central Pattern Generator (CPG) scheme to generate and control its locomotion patterns [9–11]. A plausible CPG based neural controller was then designed, able to generate the joint signals and the consequent stepping diagrams for the fruit fly. The designed network was used to control an artificial model of the fruit fly built using a dynamic simulation environment as will be reported in the next sections.

In literature several CPG-based central structures were developed and applied to different robotic platforms [12]. Here the possibility to host signals coming from sensors can improve the robot performance in terms of adaptability to the environment state [13]. The use of dynamical oscillators is also commonly exploited to represent the overall joint activity of a whole neural group and the different topological links among the oscillators give the opportunity to develop a rich variety of robot behaviors [14]. The various locomotion gaits are obtained imposing different phase displacements among the oscillators, which however, have to maintain in time the imposed phase synchronization. Adaptive walking and climbing capabilities were also obtained in real and simulated hexapodal structure referring to CPG realised via CNN architectures [15, 16]. However, only a few works deal with the problem of stability of the obtained gait, which is indeed a crucial aspect to be analyzed. In the proposed work a network of coupled oscillators is used to control the 18 DOFs of a *Drosophila*-like hexapod structure. To create a stable gait generator, a two layer structure is used to uncouple the gait generation mechanisms from the low level actuation of the legs that present different peculiar kinematics structures. To guarantee the stability of the imposed locomotion gaits, the partial contraction theory [17] has been suitably applied. The proof of convergence to every imposed gait thanks to the particular tree structure of the proposed CNN network is guaranteed. The defined CPG is then available to control the locomotion of an hexapod simulated robot with a variety of gaits that can be obtained changing the phase relations between the interconnected neurons dedicated to each leg. In Nature the locomotion pattern is changed in time depending both on the environmental constraints and on the speed imposed by the internal state of the insect. Therefore we proposed to include, as a

higher controller a neural structure similar to a Motor Map. This neural net creates an unsupervised association between the reference speed that, together with the actual speed, is given as input, and the phase value used to synchronize the CPG neurons in order to control the robot speed.

Motor Maps were already used in a number of different complex control issues. In particular, they were used, together with CNNs, to model bio-inspired perceptual capabilities implemented on roving robots [18, 19]. An approach similar to that one presented here already appeared in previous works: there a simplified symmetric structure was considered for the robot and different controlling parameters at the level of CPG were taken into account to fulfill the task [20].

In this work the control parameters are exactly the phases among the legs that can be freely imposed without loosing the phase stability, thanks to powerful theoretical results recently found in this class of non-linear systems.

In the next sections the complexity of the problem will be explained to understand why a linear controller like a PID cannot be used to solve the proposed task.

The learning process was carried out in a dynamic simulation were a *Drosophila*-like hexapodal structure was designed. Interesting information on the real stepping diagrams of the system can be extracted from the environment together with the robot position and speed in time, in order to evaluate the performance of the developed neural controller.

5.2 The Neural Network for Locomotion Control

The CNN based locomotion controller for our bio-robot follows the traditional guidelines that characterize the Central Pattern Generator paradigm. This is divided into subnetworks. Starting from the lowest level, motor neurons and interneurons are devoted to stimulate the muscle system for each of the limbs of the animal. The neural control of the motion of each limb suitably fits the kinematic constraints and geometric parameters of the limb itself so as to evoke a set of fixed action patterns. The way in which the different limb motions are synchronized to achieve an organized locomotion activity for the animal is managed by a higher level net of Command Neurons. These generates the suitable phase displacement for the implementation of a number of different locomotion patterns, which vary according to the environmental as well as to the internal state of the animal. The overall scheme of the neural controller is reported Fig. 5.1. Where the top layer represents the command system, whereas the bottom layer accounts for the local motorneuron systems. Among the different neuron models nowadays available, the authors already had introduced a neuron model that suitably matched the CNN basic cell, including the nonlinearity [11]. Its equations are reported below:

$$\begin{cases} \dot{x}_{1,i} = -x_{1,i} + (1 + \mu + \varepsilon) y_{1,i} - s_1 y_{2,i} + i_1 \\ \dot{x}_{2,i} = -x_{2,i} + s_2 y_{1,i} + (1 + \mu - \varepsilon) y_{2,i} + i_2 \end{cases} \tag{5.1}$$

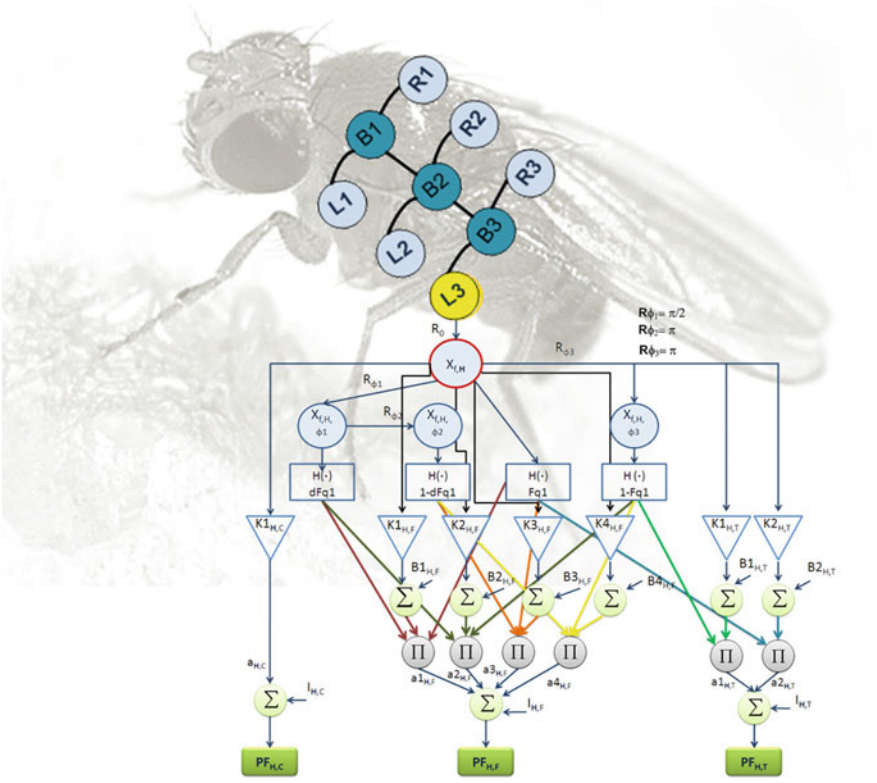


Fig. 5.1 Neural network scheme: the *top* layer generates a stable locomotion pattern, whereas the *bottom* layer is constituted by additional sub-networks generating the needed signals for the leg joints actuation

Here the authors substituted the original Piece wise linear output nonlinearity, typical of standard CNNs, with its smooth approximation $y_i = tanh(x_i)$, uniquely for simplicity in using mathematical tools for proving stability results.

By using the following parameters for each cell: $\mu = 0.23, \varepsilon = 0, s_1 = s_2 = 1, i_1 = i_2 = 0$ the cell dynamics is able to show a stable *limit cycle* behavior [21]. In this case, the μ value was chosen so as to make the ratio between the slow and the fast part of the dynamics of the limit cycle next to one, to approximate a harmonic oscillator; nevertheless, other values can be used to make the system dynamics to elicit a spiking activity.

Once defined the cell dynamics, the command network is built by locally connecting the cells using bidirectional diffusion connections. In particular, the diffusion effect implements a suitable phase shift among the command neurons, which will then become drivers for the lower level motor nets controlling each leg. To this purpose, the cloning templates for the command net can be directly defined through

rotational matrices $R(\phi)$, locally linking the command neurons. The whole dynamics configures as a two layer RD-CNN:

$$\dot{x} = f(x) - k \cdot L \cdot x \tag{5.2}$$

where x is the state variables vector $(x_1, \ldots, x_{2N})^T$, N is the number of cells; $f(x) = [f(x), \ldots, f(x_{2N})]^T$ is the dynamics of the whole uncoupled system; L is the laplacian diffusion matrix, k is the diffusion coefficient, standing for a coupling gain.

This equation, written in terms of a standard RD-CNN reads:

$$\dot{x_i} = -x_i + \sum_{Cell(j) \in N_r(i)} \left[A_{i;j} y_j + B_{i;j} u_j + C_{i;j} x_j \right] \tag{5.3}$$

$$1 \leq i, j \leq N$$

Here, being the system autonomous, $B = 0$. On the other hand, the laplacian operator modulates directly the influences among the state variables; therefore, the paradigm of the state controlled CNN, introduced in [22] is here used. Being the CNN cell a second order system it results:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}; \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}; \tag{5.4}$$

with

$$\begin{aligned} A_{11} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1+\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad A_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ A_{21} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad A_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1+\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}. \end{aligned} \tag{5.5}$$

$$C_{11} = \begin{pmatrix} 1 & \cos(\phi_{i-1;i}) & 0 \\ -\cos(-\phi_{i-1;i}) & d & -\cos(\phi_{i;i+1}) \\ 0 & -\cos(-\phi_{i;i+1}) & 1 \end{pmatrix};$$

$$C_{12} = \begin{pmatrix} 0 & -\sin(\phi_{i-1;i}) & 0 \\ -\sin(-\phi_{i-1;i}) & 0 & \sin(\phi_{i;i+1}) \\ 0 & \sin(-\phi_{i;i+1}) & 0 \end{pmatrix};$$

$$C_{21} = \begin{pmatrix} 0 & \sin(\phi_{i-1;i}) & 0 \\ \sin(-\phi_{i-1;i}) & 0 & \sin(\phi_{i;i+1}) \\ 0 & \sin(-\phi_{i;i+1}) & 0 \end{pmatrix};$$

$$C_{22} = \begin{pmatrix} 1 & \cos(\phi_{i-1;i}) & 0 \\ -\cos(-\phi_{i-1;i}) & d & -\cos(\phi_{i;i+1}) \\ 0 & -\cos(-\phi_{i;i+1}) & 1 \end{pmatrix}. \tag{5.6}$$

The parameter d is equal to the number of cells directly connected to the considered one; this corresponds to the un-weighted degree of the underlying graph. Moreover zero boundary conditions were considered. The template parameters can be easily derived considering that, in view of the bidirectional connections, among a given cell and the neighbors, there exists a precise phase displacement ϕ which is imposed using the classical rotation matrix in $\mathbf{R}^2$:

$$R(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix}; \tag{5.7}$$

More details can be found in [1].

Being the connection matrix L defined as a function of the imposed phase shift among the oscillators, L imposes a particular locomotion pattern through the associated *Flow Invariant Subspace M* [17], which was proven to be a global exponential attractor for the network dynamics. In fact the particular topology of the RD-CNN, used as the command neuron net, can be seen as a dynamic undirected diffusive tree-graph consisting of 9 neurons (see Fig. 5.1). For this particular family of configurations, an important result on the global asymptotic stability was derived [23]: any desired phase shift among the cells can be obtained if the following constraint is imposed for the diffusion coefficient k:

$$k \cdot \lambda_1 > \sup_{x_i,t} \lambda_{max} \left(\frac{\vartheta f}{\vartheta x}(x_i, t) \right) \tag{5.8}$$

where λ_1 is the algebraic connectivity of the graph associated to the network [17]. This guarantees asymptotic phase stability to the network, i.e. any desired phase among the command neurons (which will reflect into a phase shift among the robot legs) can be obtained. Once the topology is fixed (in terms of cell structure and network tree topology), all the parameters in Eq. 5.8 are known: the suitable k value can be therefore selected so as to make the network converge exponentially to any arbitrary flow invariant subspace M, defined through the phase displacements [24, 25].

5.2.1 Leg Motor Neuron Network

Neural signals, consisting in oscillating potentials from the command neuron net, reach the lower level neural structures innervating each of the limbs. These neurons have to elicit fixed action patterns synchronized with the wave of neural activity imposed by the command neurons. It is known that in many insects, including adult

Drosophila, the six legs move in a highly coordinated way, thanks to a network of axons coming from the part of the central nervous system located in the thoracic ganglia and synapsing onto specific muscle set [26]. The motor neuron network that we are presenting here consists of a series of neurons which are enslaved by the command neurons and send combined signals to the leg actuators. These signals are peculiarly shaped so as to adapt to the particular leg kinematic structure. The particular motor control network designed for each leg can be still considered as a CNN, since all neurons are characterized by local connectivity and retain the same structure as in Eq. 5.1. One main difference over the command neuron net is that connections among the motor neurons are mono directional. This gives the possibility to act in a top down fashion and prevents disturbances acting at the bottom layer to reach and affect the overall command neuron dynamics. This functional polarity is common of synapses and frequently met in locomotion control of multi legged systems, endowed with both chemical and electrical rectifying synapses [27]. This is really useful for our purposes, since a leg could be even temporarily disconnected from the command system without affecting the high level organized dynamics. This can be necessary to perform particular steering maneuvers (like turning on the spot) or special strategies for looking for a suitable foot bold position. An example of the neural motor system designed for the front leg of our robot prototype is depicted in the bottom part of Fig. 5.1. The state variables of the motor neurons are also post-processed and combined through gating functions, gains, offsets and multipliers to provide the appropriate *Primitive functions* controlling the coxa, femur and tibia joints for each leg. For the case of the rear leg, all neuron oscillators have the same frequency. For an accurate implementation of the motions for the middle and front leg, the presence of cells oscillating at a frequency resulting the double with respect to the one adopted for all the others is needed. In this case a specific control strategy based on impulsive synchronization has been implemented [28]. The network designed, in spite of its apparent complexity, can allow a high degree of adaptability by modulating a small set of parameters.

5.3 Reward-Based Learning for Speed Control

The RD-CNN structure presented above does not show any aspect related to learning or adaptation. However a highly degree of adaptability is required for a flexible locomotion control. In this paper we refer on how to introduce a suitable strategy to modulate the robot velocity. The insect brain computational model recently designed [1], hypothesizes the presence of two main blocks: the *Decision layer* and the *Motor layer*, which includes the *Description of behaviors*. The former, according to specific drives coming from the internal state of the animal or from specific external inputs, selects the particular behavior to be taken, whereas the latter is in charge for the description of the behavior to be implemented in terms of the consequent motor organization at the level of the limbs. Whereas some of the most basic behaviors are inherited, some others have to be learned to face with novel circumstances. In

our specific case of learning speed control, we can assume that the drives impose a specific speed reference value and this has to be translated into a particular locomotion pattern that satisfies the control needs. Indeed speed control in hexapod is achieved modulating both the oscillation frequency of the neural control units and the phase displacement among the legs. In this work we refer only to the latter strategy which efficiently produces a modulation of the robot speed. In addition, it is required that the learning of phase displacement should take place in an unsupervised manner. To this aim, a particular neural network, known as *Motor Map*, was used. This is a generalization of the Self Organizing Feature Maps introduced by Kohonen. Here specific characteristics of the input patterns are mirrored in specific topological areas of the responding neurons: the space of the peculiar input pattern feature is mapped into the spatial location of the corresponding neural activity [29, 30]. This interesting potentiality can be well exploited for motion pattern generation, leading to the introduction of the *Motor Maps* (MMs) [31, 32]. Here the location of the neural activity within the Kohonen layer is able to produce a trainable weighted excitation, able to generate a motion which best matches an expected *reward signal*. Two layers are so considered: the Kohonen layer, devoted to the storage of learnable input weights and giving rise to a self organized topographic map, and the motor layer, where trainable output weights associate a suitable control signal to each input. The plastic characteristics of the Kohonen layer should also be preserved in the assignment of output values, so the learning phase deals with updating both the input and the output weights.

This is an extension of the winner-take-all algorithm. Once defined the dimension of the topographic map, typically by a trial and error method, and once randomly initialized the input and output weights, a Reward function is defined, which will guide the overall learning phase. At each learning step, the neuron q which best matches the pattern given as input is selected as the winning neuron.

Its output weight is used to perform the following *perturbed* control action A_q:

$$A_q = w_{q,out} + a_q \lambda \tag{5.9}$$

where $w_{q,out}$ is the output weight of the winner neuron q, a_q is a parameter determining the mean value of the search step for the neuron q, and λ is a Gaussian random variable with a zero mean. This is a way to guarantee a random search for possible solutions. Then the increase for the delta Reward Function (*DRF*) is computed and, if this value exceeds the average increase b_q gained at the neuron q, the weight update is performed; otherwise this step is skipped. The mean increase in the reward function is updated as follows:

$$b_q(new) = b_q(old) + \rho(DRF - b_q(old)) \tag{5.10}$$

where ρ is a positive value. Moreover, a_q is decreased as more and more experience is gained (this holds for the winner neuron and for the neighboring neurons), according to the following rule:

$$a_i(new) = a_i(old) + \eta_a \xi_a (a - a_i(old)) \tag{5.11}$$

where i indicates the generic neuron to be updated (the winner and its neighbors), a is a threshold the search step should converge to, and η_a is the learning rate, whereas ξ_a takes into account the fact that the parameters of the neurons to be updated are varied by different amounts, defining the extent and the shape of the neighborhood. If $DRF > b_q$, the weights of the winner neuron and those of its neighbors are updated following the rule:

$$\begin{cases} w_{i,in}(new) = w_{i,in}(old) + \eta\xi(v - w_{i,in}(old)) \\ w_{i,out}(new) = w_{i,out}(old) + \eta\xi(A - w_{i,out}(old)) \end{cases} \tag{5.12}$$

where η is the learning rate, ξ, v, w_{in}, and w_{out} are the neighborhood function, the input pattern, the input weights and the output weights, respectively. The subscript takes into account the neighborhood of the winner neuron.

The steps involving Eqs. 5.9–5.12 are repeated. If one wishes to preserve a residual plasticity for a later re-adaptation, by choosing $a \neq 0$ in Eq. 5.11, learning is always active.

The idea proposed in this work is to use a hybrid approach joining the real time computation of RD-CNNs for generating stable locomotion patterns and a Motor Map as a high layer controller on the CNN-CPG, at the aim to control the speed of the robot, adapting the phase coordination among the legs. The MM input layer receives both the actual speed of the robot and the target speed used to evaluate the actual error. Each neuron provides a control law that modifies the phase displacement between the legs of the system.

The strategy adopted for the unsupervised modulation of the phases starts from considering the three stereotyped gaits generally adopted by hexapods and reported, in terms of phase displacements, in Table 5.1. These vary around the tripod gait and are able to maintain both static and dynamic stability. Here they are implemented by considering the front left leg $L1$ as the reference leg and imposing a fixed phase relation between the front legs ($\phi_{L1,L1} = 0°$; $\phi_{L1,R1} = 180°$) for any of the selected gaits. From the inspection of Table 5.1 it emerges that, for example, a migration from gait G1 to G2 would imply a variation $\delta\phi_{L1,L2} = \delta\phi_{L1,R2} = -30°$ and a variation $\delta\phi_{L1,L3} = \delta\phi_{L1,R3} = -60°$. Moreover, referring to Fig. 5.1, the network of command neurons was designed to leave the possibility to impose the oscillation phase of each leg independent on that one of the others. This is also allowed from the theoretical results, discussed previously, which enable any imposed phase dis-

Table 5.1 Phase relation within legs in stereotyped locomotion gaits

Gait type	Speed (bodylength/s)	$\phi_{L1,L1}$	$\phi_{L1,R1}$	$\phi_{L1,L2}$	$\phi_{L1,R2}$	$\phi_{L1,L3}$	$\phi_{L1,R3}$
G1	0.6	0	180°	270°	90°	180°	360°
G2	0.73	0	180°	240°	60°	120°	300°
G3	0.77	0	180°	180°	0°	0°	180°

Table 5.2 Parameters used for the motor map-based architecture designed to solve a speed control task

Parameters	Value
λ	$[-40, 40]$
η	$0.2 + 0.8 * e^{(-0.6t)}$
a_q	0.8
b_q	0
a_{thr}	0.2

placement among the legs to be reached exponentially. Of course, phase stability does not imply the dynamic stability of the gait. For this reason we preferred to move within the set of stable gaits $G1$, $G2$, $G3$, allowing the MM to find the most suitable phases to attain the desired speed. Within this plethora of gaits it is envisaged that a given phase imposed to a specific leg does not affect the phase associated with any other leg; to this aim the neurons belonging to the central backbone are synchronized and the phase displacements are imposed only to the connections from the backbone neurons to the outer cells. In this way, referring to Table 5.1, it holds, for example: $\phi_{L1,R2} = \phi_{B2,R2}, \phi_{L1,L3} = \phi_{B3,L3}$, and so on. This symmetry in the phase modulation between the right and left side of the structure leads to adopt only two output trainable weights for the MM. Also the MM acts by adding increments in the phase displacement among the legs: this is preferable over sharply imposing absolute phases, that could imply potential loss of stability. In details, one of the two output weights represents the phase modulation $\delta\phi_{B2,L2}$, (which will also be imposed to $\delta\phi_{B2,R2}$ for symmetry), the other stands for $\delta\phi_{B3,L3} = \delta\phi_{B3,R3}$. To find the suitable output weights a reward function that takes into account the speed error is considered and used to guide the learning process: $R = -(speed_{actual} - speed_{ref})^2$. The learning phase will find the weight set leading R as much as possible close to zero. The MM parameters used in the following simulations are reported in Table 5.2.

5.4 Dynamic Simulator

To validate the approach proposed, a dynamic simulator represents a suitable platform where these cognitive bio-inspired structures can be simulated, in view of being implemented in real robot prototypes for real life scenarios. The simulator is written in C++, and uses Open Dynamics Engine (ODE) as a physics engine to simulate dynamics and collision detection, and Open Scene Graph (OSG) as high performance 3D rendering engine [1, 33]. The main novelty of this approach consists in the extreme extensibility to introduce models. In fact, to import robot models in the simulator, a procedure was developed which starts from models realized in 3D Studio MAX and provides, using NVIDIA Physics Plugin for 3D Studio MAX, a COLLADA (COLLAborative Design Activity) description of the model to permit the correct transport in the simulated environment. In this way, the possibility to simulate own

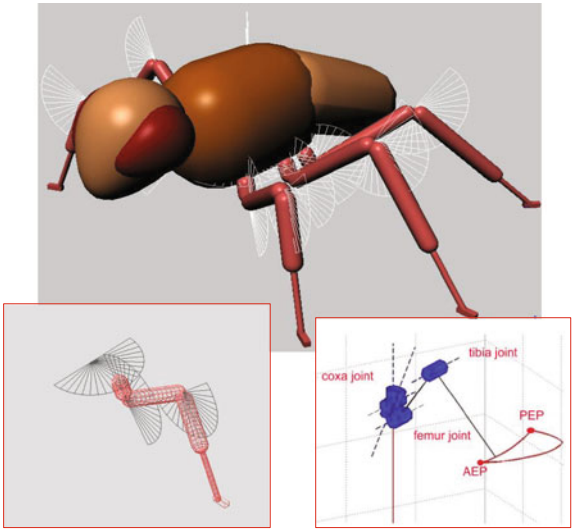


Fig. 5.2 Dynamical model of the *Drosophila*-inspired hexapod robot; the *lower left* panel reports the details of the operation range for the three joints of the *left front* leg, whereas the *right bottom* panel reports the kinematic simulation of the hind leg, showing the motion of the feet, cycling among the Anterior Extreme Position (AEP) and the Posterior Extreme Position (PEP)

environments and robots is guaranteed. The dynamical model of the *Drosophila*-inspired robot is shown in Fig. 5.2 where the typical asymmetric design and sprawled posture is evident. The structure includes a total of 18° of freedom, three for each leg. The legs, like in the real insect, are different in shapes and functionalities. Figure 5.2 reports also the operation ranges for the different leg joints for the front leg, as well as the rotation axes for the leg actuators, drawn by the corresponding Matlab Kinematic simulation. Here the trajectory spanned by the tip of the hind leg is also reported. The emerging limit cycle is the result of the application of the three *Primitive Functions* arising by the motor neuron net in Fig. 5.1. The robot is equipped with distance sensors, placed on the head and ground contact sensors located on the tip of each leg. The dimension is in scale with the biological counterpart with a body length of about 2.5 mm.

A block diagram showing the complete control structure is reported in Fig. 5.3. The CPG is able to generate the locomotion patterns of the robot depending on the control signals coming from the Motor Map for the speed control and from a reflexive behavior path that can trigger obstacles avoidance behaviors if an object is detected from the sensors equipped on the robot.

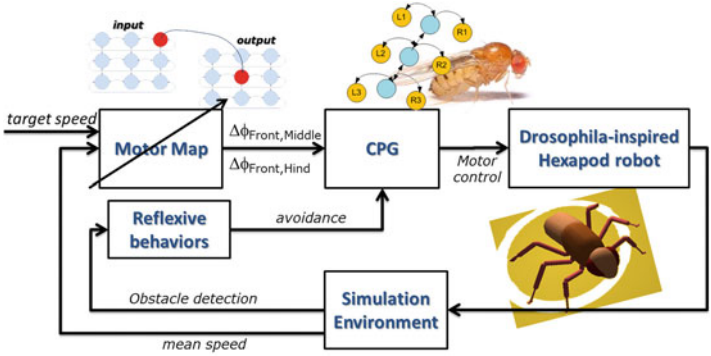


Fig. 5.3 Block diagram of the control system used to guide the locomotion of the hexapod structure. The CPG motor signals are modulated by the inputs provided by the Motor Maps for speed control and by the reflexive path that takes the lead in presence of obstacle to be avoided. The MM receives in input the reference speed and the current speed whereas the reflexive behavior block is elicited in presence of an obstacle detected through distance sensors. The MM acts on the CPG by changing the phase displacement of the middle and hind legs whereas the reflexive block inhibits the learning procedure for the MM and activates a turning strategy

5.5 Simulation Results

Analyzing the learning process in the proposed architecture we considered to start from gait $G2$ as initial configuration. The input related to the target speed, used for the MM neurons, is important when a time varying speed profile should be followed. For a first trial we considered a fixed speed, limiting the input to the speed error. This solution allowed to use a reduced number of neurons to create the map. In the following simulations a $3x3$ lattice was adopted in the Kononen layer of the MM. To evaluate the mean speed of the robot in the dynamic simulator, soon after imposing a new phase configuration, a complete stepping cycle is performed to leave the dynamics to reach a steady state: then the displacement over the two subsequent stepping cycles is evaluated to have a consistent speed result.

The stepping time imposed by the CPG to each leg is about 1.5 s. The arena used during the learning process is limited by walls. When the robot detects an obstacle (or the arena walls) with the distance sensor placed on the head, a turning strategy is applied. During this avoidance behavior the speed evaluation is stopped and the learning process waits until the procedure is completed.

The time evolution of the input and output weights after an initial transient is shown in Fig. 5.4. It can be noticed that each neuron specializes to incrementally reach the desired solution: the topological arrangement of the Kohonen layer leads to the specialization of each neuron to a specific range of the input value. The output weights affect the leg phase displacement and the trend is shown in Fig. 5.5 where the anterior left leg ($L1$) that is directly connected with the backbone with zero phase is considered as reference for all the other legs. The phase relation for the anterior

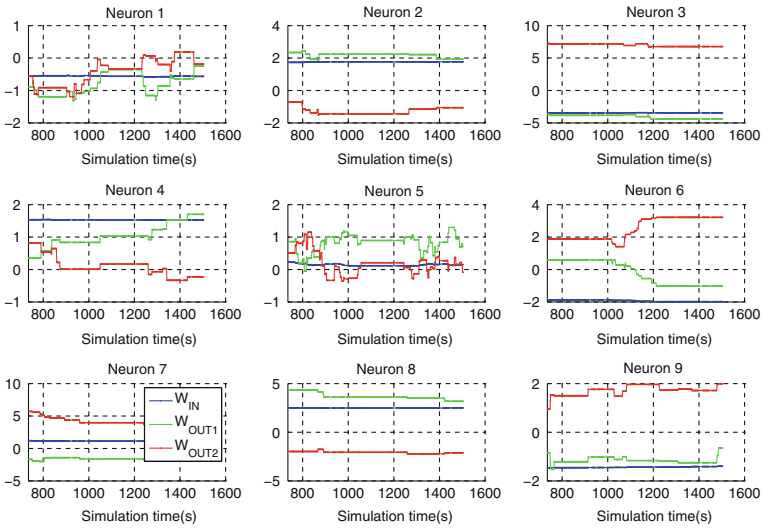


Fig. 5.4 Time evolution of the input and output weights of the MM after an initial transient

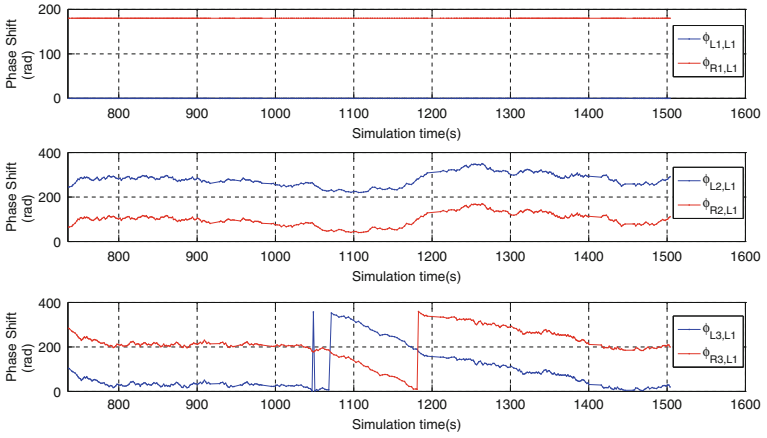


Fig. 5.5 Phase displacement in middle and hind legs obtained during learning

legs is not affected by adaptation, as explained above, and remains unchanged during the learning process.

The performance of the robot when autonomously learning to follow the reference speed is reported in Fig. 5.6. The initial speed of about 1 bodylength/s changes in time to reach a steady state solution of about 0.57 bodylength/s that was provided as reference speed.The same figure reports also the trend of the input weights related to each of the Kohonen neurons. Their values, after an initial transient, reach their

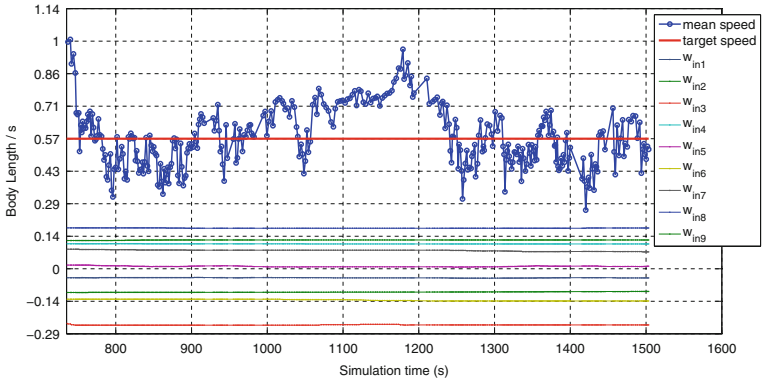


Fig. 5.6 Time evolution of the input weights and the robot speed. The target speed is 0.57 bodylength/s and the robot after a transient converges toward this value. Each *circle* indicates a measured speed, the distance between each marker is not constant because the speed evaluation is stopped during the obstacle avoidance behavior

steady state distribution which encodes the unsupervised, topological clustering of the input velocity space.

The locomotion gait can be analyzed using the stepping diagram that reports the stance and swing phase for each leg. The experiment reported in Fig. 5.7 shows a typical simulation starting from the $G2$ gait configuration: during learning, the imposed phase from the MM controller onto the legs results in a gait transition, appreciated through the modulation of the stepping diagram of the robot. This diagram is recorded acquiring information from the ground contact sensor located on the tip of each leg. In this way it is possible to better understand the phase relation between legs, also taking into account the noise intrinsically present in a dynamic environment. Learning causes the phase locking between legs to change in time, searching for a suitable configuration that matches the desired speed value. It should be noticed that the application of the proposed strategy to a dynamic simulator is really similar to the outcome in a real experiment. Learning the complex map between the error speed and the corresponding phase displacement to be imposed passes through a series of unsuccessful trials where, for a given speed error currently provided in input, a trial phase is applied to the leg: if this choice is a failure, i.e. it does not contribute to an increase in the reward function, the simulator cannot come back to the previous stage. Instead it continues with the applied phase looking for a future rewarding choice.

An important element to evaluate the system performance is to test the control architecture: after $1,500s$ of simulation (i.e. in this time window the learning cycles in the motor map oscillate between 200 and 300) the weight adaptation was frozen and the network performance was tested. An interesting test is shown in Fig. 5.8, where a time varying speed profile is given as target to the robot.The robot easily reaches the first assigned speed of 0.71 bodylength/s and, in a short time, is able to

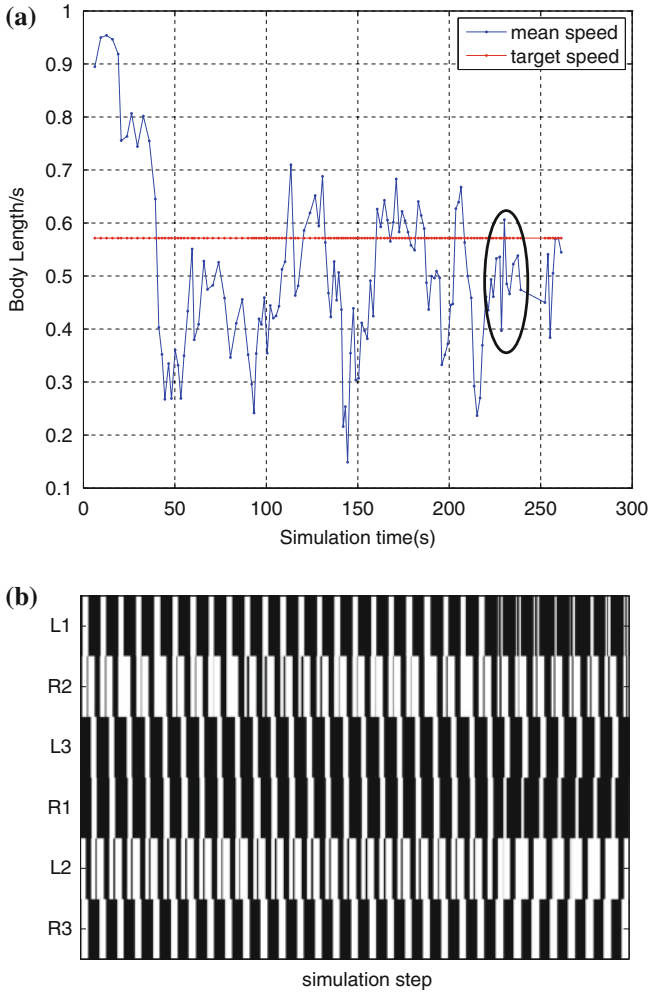


Fig. 5.7 Stepping diagram obtained looking to the ground contact sensors placed in each leg tip of the simulated *Drosophila*-inspired robot. **a** The system starts from a phase configuration next to the gait $G2$ and evolves in a series of free gaits trying to reach the desired speed. **b** In the stepping diagram that corresponds to the area outlined in the panel **a** the stance phase is shown in *black*, the swing phase in *white* and the legs are labeled as *left* (L) and *right* (R) and from *front* to *back* with numbers

readapt the robot gait to reduce the speed to 0.57 bodylength/s to finally grow up again to the previous reference speed. The phase adaptation in time is also shown: during the testing phase the leg phases change due to the incremental effect of the output weights that however remain unchanged in time. As a classical controller, the learned non linear control law is governed by the varying speed error, provided at the network input.

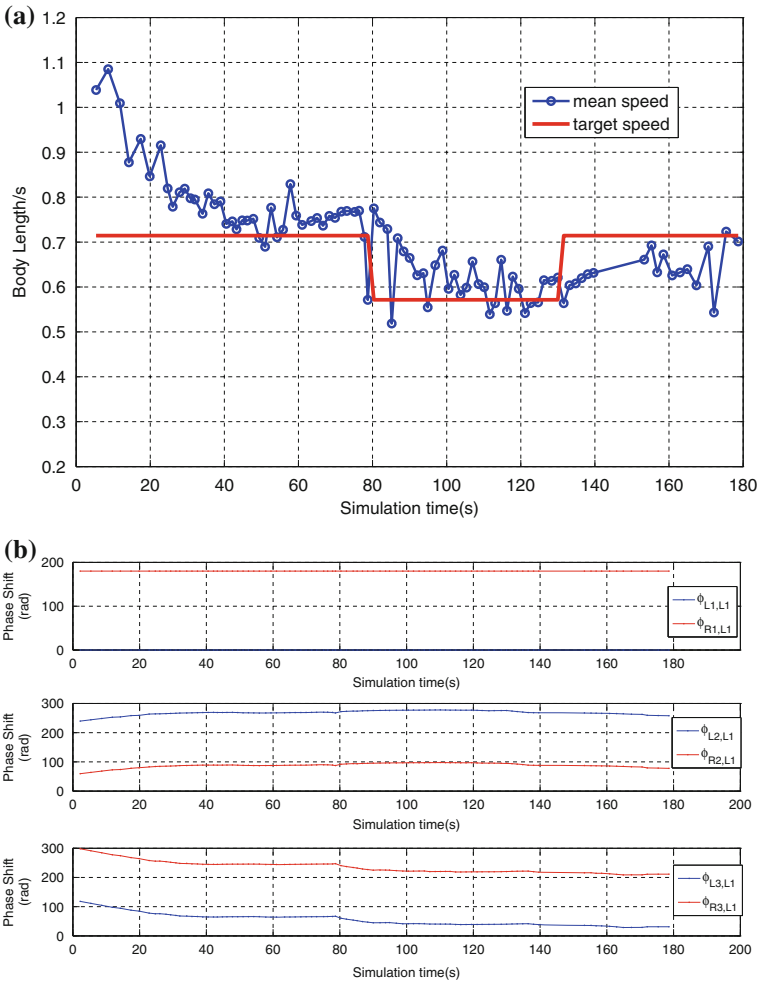


Fig. 5.8 Testing phase with different target speed values. **a** Time evolution of the robot speed. The target speed changes from 0.71 to 0.57 bodylength/s and the robot after a transient, converges toward the reference value in time. **b** Trend of the phase displacement among the robot legs during the testing phase

The learning is robust also to different initial configurations of the system walking gait (see Table 5.1). Using the same network already learned as previously presented, the robot is able to reach the target speed also starting from a different gait (i.e. *G*2 gait) not used as starting condition during the learning iterations. Figure 5.9 shows the followed speed profile that converges to a target value of 0.57 bodylength/s.

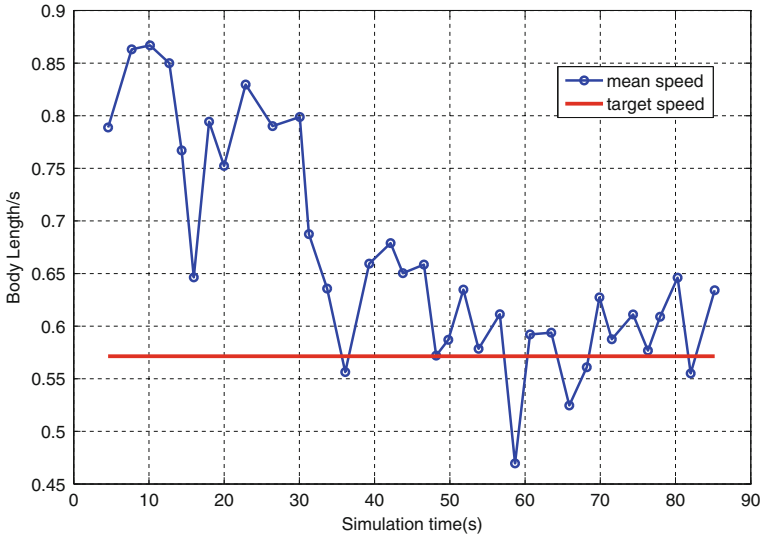


Fig. 5.9 Time evolution of the robot speed. The target speed is 0.57 bodylength/s and the robot starts from a medium gait configuration that differs from the learning phase where a slow gait was considered as starting locomotion pattern

At the aim to evaluate the efficiency of the learned control law, the a-posteriori evaluation of the searching space in the phase domain is significant to enhance the important role of the MM speed controller. Figure 5.10 gives a qualitative idea about the searching space where the different phase configurations of the legs are related with the associated robot speed. The searching domain is complex and a non-linear mapping provided by the MM is needed to find a suitable solution to solve the speed control problem. To better illustrate the movement on the map the trajectory (in the space of the imposed phases) is also reported. Starting from a given initial configuration, the robot reaches the target speed finding the suitable path within the highly non-linear searching space.

5.6 Conclusions

Modeling neural structures is an important link in a chain connecting together Biologists and Engineers that cooperate to improve the knowledge on the neural mechanisms generating our behaviors and to develop new robots able to show adaptive capabilities similarly to their biological counterparts. A cellular automata approach to this problem, extended to the analog time domain with the CNNs, gives an interesting prospective because it easily allows theoretical analysis concerning stability issue, and also thanks to the mainly local connections between neurons, it creates a short-cut to the hardware implementation: different solutions are available using

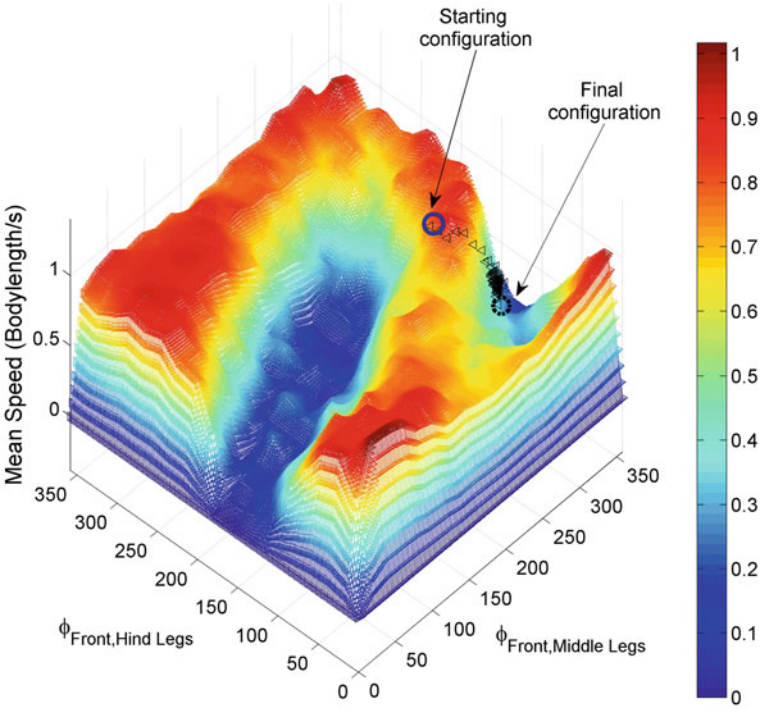


Fig. 5.10 Map of the searching space where, for each phase pair for middle and hind legs, the corresponding speed value is considered (i.e. reported in the z-axis). The trajectory followed by the system in the phase domain during a testing simulation is also reported: the starting point speed is about 0.89 bodylength/s, the robot reduced it reaching a speed of 0.46 bodylength/s following the learned trajectory in the phase space thanks to the MM neurons action

both micro-controller and dedicated integrated circuits. A Central Pattern Generator has been designed to generate the locomotion patterns for an asymmetric hexapod robot inspired by the *Drosophila melanogaster*. The integration of a neural controller based on Motor Maps allowed to adaptively control the robot speed acting on the CPG parameters. The reported results showed that the dynamically simulated robot was able to follow a desired speed profile incrementally adapting the phase displacement between legs. The neural controller shows interesting generalization capabilities to efficiently respond to novel initial conditions and time-varying speed profiles. This approach to adaptive speed control has to be considered as a brick within the much wider design of an insect brain computational model, where adaptive locomotion capabilities are required to show complex cognitive skills for the next generation of adaptive intelligent machines more and more mimicking their biological counterpart.

Acknowledgments This work was supported by EU Project EMICAB, grant no. 270182.

References

1. Arena, P., Patané, L. (eds.): Spatial temporal patterns for action-oriented perception in roving Robots II. In: An Insect Brain Computational Model Springer Series, Cognitive Systems Monographs, vol. 21, Springer, Berlin (2014)
2. Arena, P., Patané, L. (eds.): Spatial Temporal Patterns for Action-Oriented Perception in Roving Robots. In: Series, Cognitive Systems Monographs, vol. 1, Springer, Berlin (2009)
3. Arena, P., Fortuna, L., Lombardo, D., Patané, L.: Perception for action: dynamic spatiotemporal patterns applied on a roving robot. Adapt. Behav. **16**(2–3), 104–121 (2008)
4. Arena, P., Fortuna, L., Frasca, M., Lombardo, D., Patané, L., Crucitti, P.: Turing patterns in RD-CNNS for the emergence of perceptual states in roving robots. Int. J. Bifurcat. Chaos **17**(1), 107–127 (2007)
5. Arena, P., Fortuna, L., Lombardo, D., Patané, L., Velarde, M.G.: The winnerless competition paradigm in cellular nonlinear networks: models and applications. Int. J. Circuit Theory Appl. **37**(4), 505–528 (2009)
6. Arena, P., De Fiore, S., Fortuna, L., Nicolosi, L., Patané, L., Vagliasindi, G.: Visual Homing: experimental results on an autonomous robot. In: Proceedings of 18th European Conference on Circuit Theory and Design (ECCTD 07) Seville, Spain (2007)
7. Arena, P., De Fiore, S., Patané, L., Termini, P.S., Strauss, R.: Visual learning in Drosophila: application on a roving robot and comparisons. In: Proceedings of 5th SPIE's International Symposium on Microtechnologies Prague, Czech Republic (2011)
8. Mronz, M., Strauss, R.: Visual Motion integration controls attractiveness of objects in walking flies and a mobile robot. In: Proceedings of International Conference on Intelligent Robots and Systems, pp. 3559–3564, Nice, France (2008)
9. Arena, P., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: CPG-MTA implementation for locomotion control. In: Proceedings of IEEE International Symposium on Circuits and Systems, pp. 4102–4105 (2005)
10. Arena, P., Fortuna, L., Frasca, M., Patané, L., Pollino, M.: An autonomous mini-hexapod robot controlled through a CNN-based CPG VLSI chip. In: Proceedings of 10th IEEE International Workshop on Cellular Neural Networks and their Applications, pp. 401–406, Istanbul, Turkey (2006)
11. Arena, P., Fortuna, L., Frasca, M., Patané, L.: Sensory feedback in CNN-based central pattern generators. Int. J. Neural Syst. **13**(6), 349–362 (2003)
12. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Networks **21**, 642–653 (2008)
13. Arena, P., Patané, L.: Simple sensors provide inputs for cognitive robots. IEEE Instrum. Meas. Mag **12**(3), 13–20 (2009)
14. Steingrube, S., Timme, M., Worgotter, F., Manoonpong, P.: Self-organized adaptation of a simple neural circuit enables complex robot behaviour. Nature Phys. **6**, 224–230 (2010)
15. Pavone, M., Arena, P., Fortuna, L., Frasca, M., Patané, L.: Climbing obstacle in bio-robots via CNN and adaptive attitude control. Int. J. Circuit Theory Appl. **34**(1), 109–125 (2006)
16. Arena, P., Fortuna, L., Frasca, M.: Attitude control in walking hexapod robots: an analogic spatio-temporal approach. Int. J. Circuit Theory Appl. **30**(2–3), 349–362 (2002)
17. Seo, K., Slotine, J.E.: Models for Global Synchronization in CPG-based Locomotion. In: Proceedings of IEEE ICRA (2007)
18. Arena, P., De Fiore, S., Patané, L.: Cellular nonlinear networks for the emergence of perceptual states: application to robot navigation control, Neural Networks **22**(5–6), 801–811 (2009). ISSN 08936080 (special issue: advance in neural networks research)
19. Arena, P., Fortuna, L., De Fiore, S., Frasca, M., Patané, L.: Perception-action map learning in controlled multiscroll system applied to robot navigation. Chaos **18**(043119), 1–16 (2008)
20. Arena, P., Fortuna, L., Frasca, M., Sicurella, G. (eds.) : An adaptive. Self-organizing dynamical system for hierarchical control of bio-inspired locomotion. Syst. Man Cybern. Part B **34**(4), 1823–1837 (2004)

21. Frasca, M., Arena, P., Fortuna, L.: Bio-inspired emergent control of locomotion systems, p. 48. World Scientific, Singapore (2004) (Serie A)
22. Manganaro, G., Arena, P., Fortuna, L.: Cellular neural networks: chaos, complexity, and VLSI processing. In: Advanced Microelectronics, vol. 1, Springer, Berlin (1999)
23. Arena, E., Arena, P., Patané, L.: Efficient hexapodal locomotion control based on flow-invariant subspaces. In: Proceedings of 18th World Congress of the International Federation of Automatic Control (IFAC), Milan, Italy (2011)
24. Arena, E., Arena, P., Patané, L.: Frequency-driven gait control in a central pattern generator. In: Proceedings of 1st International Conference on Applied Bionics and Biomechanics ICABB-2010, Venice, Italy (2010)
25. Arena, E., Arena, P., Patané, L.: Modelling stepping strategies for steering in insects. In: Proceedings of XXI Italian Workshop on Neural Networks (2011)
26. Baek, M., Mann, R.S.: Lineage and birth date specify motor neuron targeting and dendritic architecture in adult Drosophila. J. Neurosci. **29**(21), 6904–6916 (2009)
27. Edwards, D.H., Heitler, W.J., Krasne, F.B.: Fifty years of a command neuron: the neurobiology of escape behavior in the crayfish. Trends Neurosci. **22**(4), 153–161 (1999)
28. Arena, E., Arena, P., Patané, L.: CPG-based locomotion generation in a Drosophila inspired legged robot. In: Proceedings of Biorob 2012, Roma, Italy (2012)
29. Kohonen, T.: Self-organized formation of topologically correct feature maps. Bio. Cybern. **43**, 59–69 (1972)
30. Arena, P., De Fiore, S., Fortuna, L., Frasca, M., Patané, L., Vagliasindi, G.: Reactive navigation through multiscroll systems: from theory to real-time implementation. Auton. Robot. **25**(1–2), 123–146 (2008)
31. Ritter, H., Schulten, K.: Kohonen's self-organizing maps: exploring their computational capabilites. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 1, San Diego, CA, 109–116 (1988)
32. Schulten, K.: Theoretical biophysics of living systems. http://www.ks.uiuc.edu/Services/Class/PHYS498TBP/spring2002/neuro_book.html (2002)
33. Arena, P., Cosentino, M., Patané, L., Vitanza, A.: SPARKRS4CS: a software/hardware framework for cognitive architectures. In: Proceedings of 5th SPIE's International Symposium on Microtechnologies, pp. 1–12. Czech Republic, Prague (2011)

Chapter 6
Routing by Cellular Automata Agents in the Triangular Lattice

Rolf Hoffmann and Dominique Désérable

Abstract This chapter describes an efficient novel router in which the messages are transported by Cellular Automata (CA) mini-robots or so called CA agents. CA agents are compliant but inconvenient to describe with the CA paradigm. In order to implement agents more efficiently, the CA-w model (with *write* access) is used. Both CA and CA–w models are compared. The other relevant feature in this chapter is the underlying network embedded into the triangular lattice, with more symmetries, thereby providing agents with more degrees of freedom. The router uses six channels per node that can host up to six agents and provides a minimal routing scheme (XYZ–protocol). Each agent situated on a channel has a computed minimal direction defining the new channel in the adjacent node. In order to increase the throughput an adaptive routing protocol is defined, preferring the direction to an unoccupied channel. A strategy of deadlock avoidance is also investigated, from which the initial setting of the channels can be alternated in space, or the agent's direction can dynamically be randomized.

6.1 Introduction

Problem solving with robots and agents has become more and more attractive [1–6]. What are the benefits to use agents for a given problem? Generally speaking, agents are intelligent and their capabilities can be tailored to the problem in order to solve

R. Hoffmann (✉)
Technische Universität Darmstadt, FB Informatik, FG Rechnerarchitektur,
Hochschulstraße 10, 64289 Darmstadt, Germany
e-mail: hoffmann@informatik.tu-darmstadt.de

D. Désérable
Institut National des Sciences Appliquées, 20 Avenue des Buttes de Coësmes,
35043 Rennes, France
e-mail: domidese@gmail.com

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_6

it effectively, and often in an unconventional way. Important properties that can be achieved by agents are

- **Scalable**: the problem can be solved with a variable number of agents, and faster or better with more agents.
- **Tuneable**: depending on the agent's intelligence, the problem can be solved more efficiently (with a higher quality and faster).
- **Flexible**: similar or dynamically changing situations can be solved using the same agents, e.g. when the shape or size of the environment is changing.
- **Fault-tolerant**: the problem can be solved with low degradation even if some "noise" is added, e.g. dynamic obstacles or temporary malfunctions appear, or some agents break down totally.

Owing to their intelligence, agents can be employed to design, model, analyze, simulate, and solve problems in the areas of complex systems, real and artificial worlds, games, distributed algorithms and mathematical questions.

Robots or agents controlled by a finite state machine (FSM) have a long history in computer science [7], sometimes they are simply called "FSMs", often with the property that they can move around on a graph or grid. For example, searching through the whole environment by an FSM was addressed in [8] and graph exploration by FSM controlled robots was treated in [9]. In order to support the simulation of such applications special languages like [10–12] have been developed.

6.1.1 Cellular Automata Agents

What is a Cellular Automata Agent (CA Agent)? Simply speaking, a CA agent is an agent that can be modeled within the CA paradigm. And what are the most important attributes an agent should have in our context?

1. **Self-contained** (an individual, complete in itself). In CA, this property can be realized by one cell, by a part of a cell, or by a group of cells.
2. **Autonomous** (not controlled by others). Agents operate on their own and control their actions and internal states. In CA, this property can be realized by the own state and the CA rule.
3. **Perceptive** (perceives information about the environment). In CA, this property is realized by reading and interpreting the states of the neighborhood.
4. **Reactive** (can react on the perceived environment). In CA, this property is realized by changing the own state by taking into account the perceived information.
5. **Communicative** (can communicate with other agents). This property means that agents can exchange information, either indirectly through the environment (stigmergy, e.g. pheromones), or directly by perceiving other agents and reacting on them in a perceivable way.
6. **Proactive** (acts on its own initiative, not only reacting, using a plan). In CA, the cell's next state should not only depend on its neighbors' states but also on its

own state. The number of inputs and states should not be too small in order to give the agent a certain intelligence to initiate changes and to deal in advance with difficult situations. And the agent's behavior is to a certain extent not foreseeable, it can be influenced by personal secret information or internal events. In CA, this can be accomplished by hidden states that cannot be observed by the neighbors, or by asynchronous internal triggers (e.g. random generator). As it is difficult to define proactivity in a strict way, it is a matter of viewpoint whether simple classical CA rules (like Game of Life, Traffic Rule 184) shall be classified as multi-agent systems or not.

7. **Local** (acts locally). Agents are small compared to the system size and can only act on their neighborhood. Global effects arise from accumulated local actions.
8. **Mobile** (this feature is not required but very useful). Very often agents are moving around in the environment, and then the neighborhood and the place of activity are moving, too. When moving around, an agent may also change its local environment at the same time.

Usually an agent performs *actions*. *Internal* actions change the state of an agent, either a visible or a non-visible state, whereas *external* actions change the state of the environment. The environment is composed of the ground environment (constant or variable) and the other agents. In CA, an agent is not allowed to change the state of a neighboring cell. Therefore, if an agent wants to apply an external action to a neighboring cell, it can only issue a command that must be adequately executed by the neighbor. For example agent A sends a "kill" command to agent B, then agent B has to kill itself. This example shows that the CA modeling and description of changing the environment is indirect and does not appear natural. Other models are helpful to simplify such descriptions, like the "CA–w model" presented hereafter.

6.1.2 CA and CA–w Models

In order to describe moving agents, moving particles or dynamic changing activities, the CA–w model (Cellular Automata with *write* access) was introduced [13]. Simply speaking, this model allows to write information onto a neighbor. This method has the advantage that a neighbor can directly be activated or deactivated, or data can be sent actively to it by the agent. Thus the movement of agents can be described more easily.

The CA–w model is a restricted case of the more general, "*Global*" GCA–w [14–16]. In GCA–w any cell of the whole array can be modified whereas in the CA–w model only the local neighbors can be. Usually the cells of these models are a composition of (data, pointers). The neighbors are accessed via pointers, that can be changed dynamically like the data by an appropriate rule from generation to generation.

In order to avoid confusion between CA and CA–w, in this context the CA model can be attributed as "classical model" and the CA–w model as "implementation model" although both can be used for description and implementation.

What are the capabilities and limitations of CA–w compared to CA? The main difference is that the CA–w model allows to modify the state of a neighbor. Thereby the activity of a neighboring cell can be switched on or off and data can actively be moved to a neighbor, which is very useful for the description and effective simulation of active particles or moving agents. Comparing their computing power, a CA equivalent to a CA–w with neighborhood $N1$ can be found by extending $N1$ to $N2$ ($N1$ extended by write-distance). For example a CA–w with neighborhood distance 1 (read and write) is equivalent to a CA with neighborhood distance 2. Because of this equivalence, both models can be mapped onto each other.

A drawback is the possible occurrence of write conflicts. There are two solutions to handle conflicts:

- Use a conflict-resolving function, for example by applying a reduction operator (max, +, ...) or using a random or deterministic priority scheme.
- Avoid conflicts by algorithmic design, meaning that the parallel application of all rules never cause a conflict.

The second solution is more simple and elegant and many applications with agents can be described in this way. Our routing problem with agents herein is implemented by the CA–w model, although it is also possible to model a more cumbersome system by standard CA.

6.1.2.1 Modeling Agents' Mobility

How can an agent move from A to B? In the CA model, a couple of two rules (*copy*–rule, *delete*–rule) must be performed (Fig. 6.1a): the first rule copies the agent from A to B, the second deletes it on A. Both rules have to compute the same moving condition, this means a redundant computation. Two operating modes allow the CA–w to avoid this redundancy:

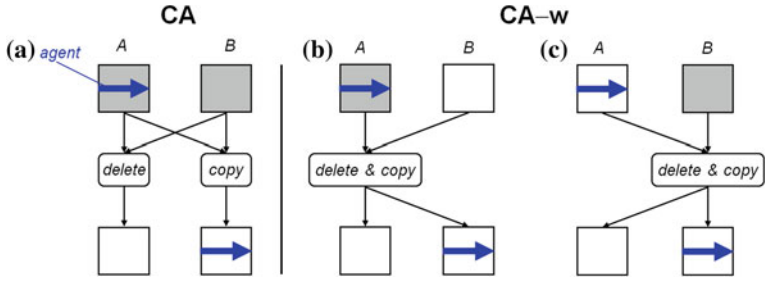


Fig. 6.1 *CA model* **a** cell A deletes the agent and cell B copies it. *CA–w model* **b** cell A deletes and copies the agent or **c** cell B deletes and copies the agent. Active cells executing a rule are shaded

- Cell A (the agent) is responsible for the moving operation (Fig. 6.1b), it computes the moving condition and, if true, applies a rule that deletes itself on A and copies it to B.
- Cell B (the empty front cell) is responsible (Fig. 6.1c), it computes the moving condition and, if true, applies a rule that deletes the agent on A and copies it to B.

The second mode is used for our problem. In this way, concurrent agents wanting to move to the same empty channel can easily be prioritized. The differences between CA–w and classical CA for our routing problem will be illustrated in Sect. 6.3.

6.1.3 Lattice Topology

Choosing the best topology for 2*d* CA agents is not straightforward. We give some insight hereafter to clarify our choice of the network used in this chapter to route agents.

6.1.3.1 Towards an Optimal Tiling

The three regular tessellations of the plane are displayed in Fig. 6.2a where Schläfli symbol $\{p, q\}$ gives an exact definition of the tiling [17]. Their associated dual $\{q, p\}$ tilings are displayed in (b), whence the three possible regular 2*d* lattices in (c), either 3–valent or 4–valent or 6–valent [18].

Two usual tessellations for 2*d* cellular automata are identified in the $\{4, 4\}$ "square" tiling and the $\{6, 3\}$ "hexagonal" tiling. It is observed that the minimal number of neighbors appears in the hexagonal tiling; moreover, the six neighboring cells are adjacent. Thereby, there is no risk of wavering as in the $\{4, 4\}$ case between either a 4–valent von Neumann neighborhood or a 8–valent Moore neighborhood. This matter of symmetry among lattices may have important impacts upon the behavior of their CA. A typical example is well known for lattice-gas automata wherein the 4–valent HPP gas cannot be consistent with the Navier-Stokes equation while the 6–valent FHP ensures consistency [19–21]. Our routing problem herein is embedded into the 6–valent lattice.

6.1.3.2 Towards an Optimal 2*d* Finite-Sized Network

Once the valence had been settled, the question is to define a finite-sized toroidal network. As a matter of fact, there is a relationship between a compatible arrangement and some associated tessellation of the plane. Tiling the $\{4, 4\}$ tessellation with a finite-sized "prototile" is examined in [22]. In general, the topologies related to plane tessellations belong to the family of multi-loop and circulant networks [23].

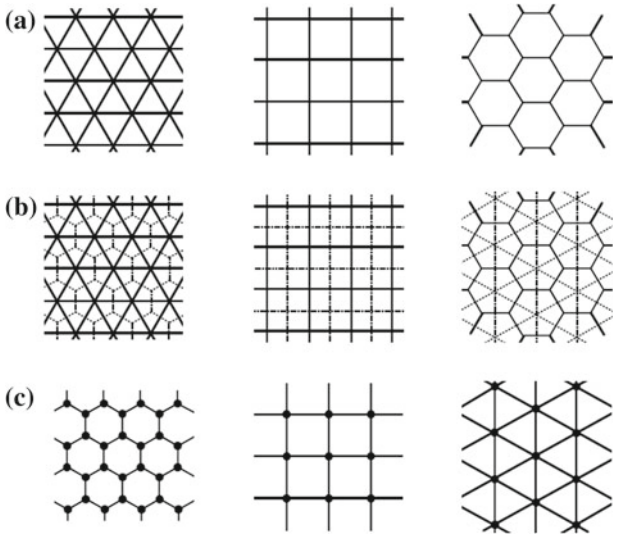


Fig. 6.2 **a** The three regular tessellations of the plane: $\{3, 6\}$ with 3–gons, $\{4, 4\}$ with 4–gons, $\{6, 3\}$ with 6–gons where $\{p, q\}$ is the Schläfli symbol; a triangular cell is surrounded with twelve neighbors, a square cell with eight neighbors, a hexagonal cell with six *adjacent* neighbors. **b** Associated dual tilings: $\{6, 3\}$, $\{4, 4\}$, $\{3, 6\}$ (in *dashed lines*); $\{q, p\}$ is the dual of $\{p, q\}$. **c** The three induced regular $2d$ lattices: *3*–valent, *4*–valent, *6*–valent

The hexagonal (or triple loop) case was investigated in [24] in order to exhibit graphs with minimum diameter. They proved that the maximum order of a triple loop graph with diameter n is $N = 3n^2 + 3n + 1$. The grid representation of the graph is a hexagonal torus with n circular rings of length $6n$ arranged around a central node. Incidentally, this family of "honeycombs" H_n was encountered elsewhere, arising in various projects such as FAIM-1 [25], Mayfly [26], HARTS [27] and more recently with the EJ networks [28].

When tiling the plane with H_n prototiles, it can be observed that the axes joining the center of these prototiles and the symmetry axes of the $\{6, 3\}$ tiling do not coincide. On the contrary, we have defined a new family of hexavalent networks stabilizing the symmetry axes, that provide these networks with the highest symmetry level for a 6–valent finite lattice. A relevant illustration of this discrepancy between prototiles can be found in [29].

6.1.3.3 Arrowhead and Diamond

Our networks belong to a family of hierarchical Cayley graphs [30]. As a consequence, this property facilitate as far as possible any routing or global communication procedure. The graphs of this family are denoted elsewhere as "*arrowhead*" or "*diamond*" in order to avoid confusion with H_n family. The reader is referred

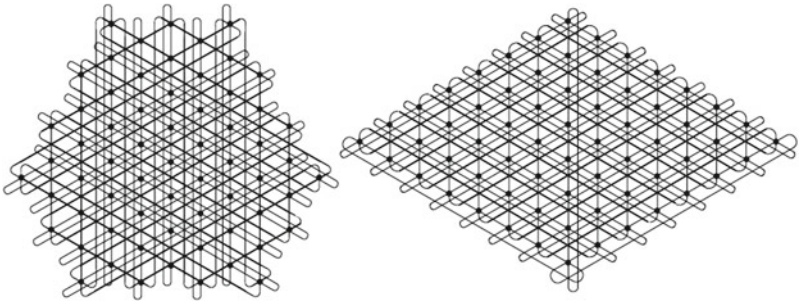


Fig. 6.3 Arrowhead and diamond with $N = 4^n$ vertices for $n = 3$

to [31] for more details about the genesis of these graphs, displayed in Fig. 6.3, and some of their topological properties. A very important one is that as Cayley graphs they are vertex-transitive, that means that any vertex behaves identically. Practically, this property involves a unique version of router code distributed among all nodes. It was also shown that these graphs provide a good framework for routing [32] and other global communications like broadcasting [33] and gossiping [34]. A survey of global communications in usual networks is given in [35] but we focus hereafter on the routing problem.

Arrowhead and diamond are isomorphic and the diamond itself is isomorphic to an orthogonal representation of the "T–grid" like in Fig. 6.5 with $n = 2$. Therefore, it is easy to map "T" into the arrowhead by a simple coordinate transformation. In the sequel, "T" –or "T_n" if the "size" n is relevant– will always denote the orthogonal representation of the diamond. It is interesting to observe that the k–ary 2–cube [36] ($k = 2^n$ herein) can be embedded into by eliminating one direction of link, namely the "diagonal" direction in the orthogonal diamond. For clarity's sake, the "S–grid" "S" –or "S_n" as well– will also denote our 2^n–ary 2–cube in the sequel as a subgrid of T. Note that another family of "augmented" k–ary 2–cubes was investigated elsewhere [37] for any k but which coincide with T_n only when $k = 2^n$.

The tori are well suited for physical ergodic systems with periodic boundary conditions [38]. For a finite space with robots or multiagents, non-periodic boundaries can also be defined with boundary conditions (bounce-back, absorption and so forth). To conclude this topological presentation, let us hope that our S–T family might reconcile von Neumann and hexagonal $2d$ cellular automata and activate exciting challenges in CA and robot worlds.

6.1.4 The Problem: Routing

Let us consider the approach based on agents transporting messages from a source node to a destination node and following a minimal route (or shortest path). The nodes are connected via twelve unidirectional links, namely two in each of the six directions,

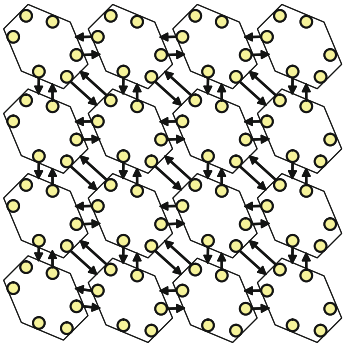


Fig. 6.4 Each node of the network contains six buffers (channels) and is connected to its neighbors by 6 input and 6 output links

that corresponds with a full-duplex or double lane traffic (Fig. 6.4). Each node is provided with six *channels* (sometimes called *buffers* according to the context) and one channel may host at most one agent transporting a message. Each agent moves to the next node, defined by the channel's position it is situated on. When moving to the next node, an agent may hop to another channel, defined by the agent's direction in its minimal path.

A *message transfer* is the transfer of one message from a source to a target *and* each agent shall perform such a message transfer. A set of messages to be transported is called *message set*. A *message set transfer* is the successful transfer of all messages belonging to the set. Initially k agents are situated at their source nodes. Then they move to their target nodes following their minimal path. When an agent reaches its target, it is deleted. Thereby the number of moving agents is reduced until no agent is left. This event defines the end of the whole message set transfer. Note that the agents hinder each other more at the beginning (due to congestion) and less when many of the agents have reached their targets and have been deleted. No new messages are inserted into the system until all messages of the current set have reached their targets. This corresponds to a barrier-synchronization between successive sets of messages. Initially each agent is placed on a certain channel (with direction to the target) in the source node and each agent knows its target. The target node of an agent should not be its source node: message transfers within a node without an agent's movement are not allowed.

The goal is to find an agent's behavior in order to transfer a message set (averaged over many different sets) as fast as possible, that is, within a minimal number of generations. We know from previous works that the agent's behavior can be optimized (e.g. by a genetic algorithm) with respect to the set of given initial configurations, the initial density of agents, and the size of the network. The goal is not to fully optimize the agent's behavior but rather to design a powerful router with six channels that outperforms the ones developed before [39, 40].

6.1.4.1 Related Work

Target searching has been studied in many variations: with moving targets [41] or as single-agent systems [42]. Here we consider only stationary targets, and multiple agents having only a local view. This contribution continues our preceding work on routing with agents on the cyclic triangular grid [39] and on non-cyclic rectangular $2d$ meshes [43]. In a recent work [40], tori S and T were compared; evolved agents, with a maximum of one agent per node, were used in both cases. It turned out that routing in T is performed significantly better than in S.

The novelty herein is that *six* agents per node are now used, with one agent per channel, instead of *one* agent per node therein. Another difference is that in [39] the agent's behavior was controlled by a finite state machine (FSM) evolved by a genetic algorithm, whereas here the behavior is handcrafted. To summarize, the goal is to find a faster router in T, using six agents per node and bidirectional traffic between nodes, at first modeling the system as CA–w and then discussing whether the routing algorithm is deadlock-free or not. Usually deterministic agents with synchronous updating are not deadlock-free. Therefore a small amount of randomness can be added to a deterministic behavior [44] in order to avoid deadlocks.

Communication protocols in hexagonal networks were already studied for H_n or EJ topologies [27, 28]. An adaptive deadlock-free routing protocol was proposed recently [45] using additional virtual channels. In our approach, if the minimal route is blocked, the alternative minimal route is attempted in order to minimize deadlocks. Further possibilities to avoid deadlocks are proposed in Sect. 6.4.3. Note that we do not address the problem of fault tolerance networks on VLSI chips [46, 47]. A general insight on adaptive routing can be found in [48].

The remainder of this chapter is structured as follows. Section 6.2 deals with the topology of the T–grid and presents the XYZ–protocol computing the minimal route. Section 6.3 shows how the routing can by modeled as a multi-agent system. An analysis of the router efficiency is discussed in Sect. 6.4 and some deadlock situations are pointed out before Summary. This work finalizes a previous one investigating this novel router in the triangular grid with six channels [49].

6.2 Minimal Routing in the Triangular Grid

6.2.1 Topology of S and T

Consider the square blocks in Fig. 6.5 with $N = 2^n \times 2^n$ nodes where n denotes the size of the networks. The nodes are labeled according to the XY–orthogonal coordinate system. In the left block, a node (x, y) labeled "xy" is connected with its four neighbors $(x \pm 1, y)$, $(x, y \pm 1)$ (with addition modulo 2^n) respectively in the W–E, N–S directions, giving the 4–valent torus S_n. In the right block, two additional links $(x - 1, y - 1)$, $(x + 1, y + 1)$ are provided in the diagonal NW–SE direction (Z–coordinate), giving the 6–valent torus T_n. Because their associated

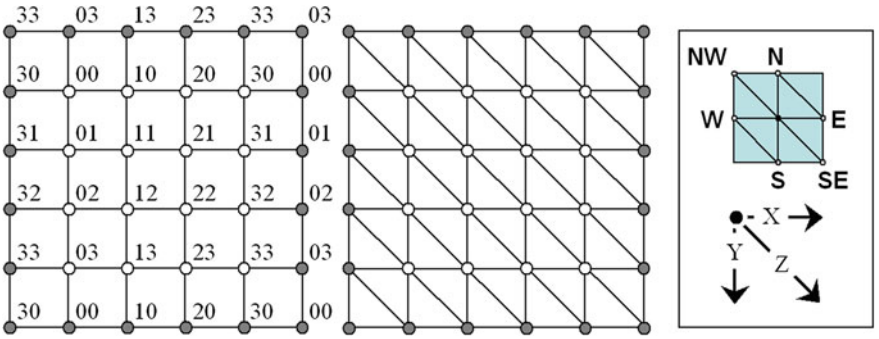


Fig. 6.5 Tori S_2 and T_2 of order $N = 16$, labeled in the XY coordinate system; redundant nodes in *grey* on the boundary. Inset: orientations W–E, N–S, NW–SE according to an XYZ reference frame

graphs are regular their number of links is, respectively, $2N$ for torus S_n and $3N$ for torus T_n. Both networks are scalable in the sense that one network of size n can be built from four blocks of size $n - 1$. The S–grid is just displayed here because it is often interesting to compare the topologies and performances of S_n and T_n, two networks of the same size.

An important parameter for the routing task in the networks is the diameter. The diameter defines the length of the shortest path between the most distant pair of nodes and provides a lower bound for routing or other global communications; such a pair is said to be *antipodal*. The exact value of the diameters in S_n and T_n is given by

$$D_n^S = \sqrt{N}; \quad D_n^T = \frac{2(\sqrt{N} - 1) + \varepsilon_n}{3} \tag{6.1}$$

where $\varepsilon_n = 1$ (resp. 0) depends on the odd (resp. even) parity of n and where the upper symbol identifies the torus type; whence the ratio denoted

$$D_n^{S/T} \approx 1.5 \tag{6.2}$$

between diameters. In this study, only the diameter D_n^T will be considered, denoted simply D_n in the sequel [50].

6.2.2 Minimal Routing Schemes in S and T

The basic, deterministic routing schemes are driven by the Manhattan distance in S [36] and by the so-called "hexagonal" distance in T [28, 32]. They are denoted as "rectangular" and "triangular" herein. Considering a source "A" and a target "B" as

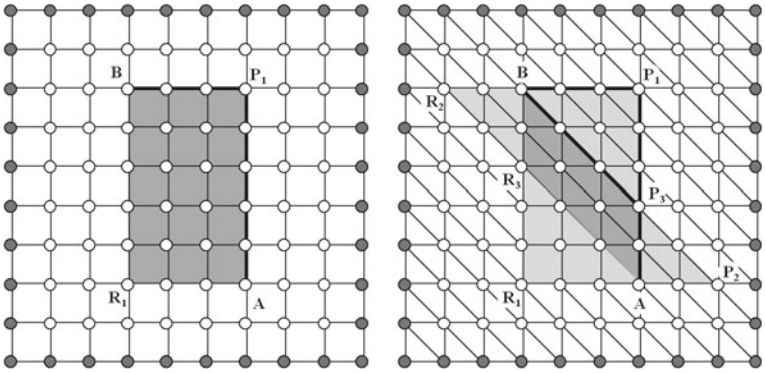


Fig. 6.6 Networks S_3 and T_3 of order $N = 64$. Routing paths from a source "A" to a target "B": rectangular routing in S (*left*), triangular routing in T (*right*). In the rectangular routing, axis systems $X_A Y_A$ and $X_B Y_B$ intersect at P_1, R_1 and yield the rectangle $A P_1 B R_1$ in general. In the triangular routing, axis systems $X_A Y_A Z_A$ and $X_B Y_B Z_B$ intersect at P_i, R_i $(i = 1, 2, 3)$ and yield three parallelograms $A P_i B R_i$ in general; in this case, the parallelogram $A P_3 B R_3$ is "minimal"

shown in Fig. 6.6, we choose to find a shortest path from A to B with *at most one change* of direction.

In the square grid on the left part, the construction yields the rectangle $A P_1 B R_1$. In order to ensure a homogeneous routing scheme, from an usual convention the agent is carried following one direction first, following the other direction afterwards. This orientation will be specified in the following subsection. Under these conditions, a route $A \to B$ and a route $B \to A$ will follow two disjoint paths and each of them is made of two unidirectional subpaths, that is $A \to P_1 \to B$ and $B \to R_1 \to A$ respectively. In a particular case, A and B may share a common axis and the routes $A \to B$ and $B \to A$ need a (full-duplex) two-lane way $A \leftrightarrow B$. Note that in a finite-sized torus, not only the "geometric" rectangle $A P_1 B R_1$ should be considered but rather a "generalized" rectangle, because the unidirectional subpaths may "cross" over the boundaries of the torus.

In the triangular grid on the right part, the construction involves three generalized parallelograms of the form $A P_i B R_i$. Among them, there exists a "minimal" one that defines the shortest path. It is the purpose of the following to detect it and to move CA agents within it.

6.2.3 Computing the Minimal Route in T (XYZ–Protocol)

The following abbreviations are used in the routing algorithm:

$sign(d) = (0, 1, -1)$ IF $(d = 0, d > 0, d < 0)$ for any integer d and

$\bar{d} = d - sign(d) \cdot M/2$, where $M = 2^n$ is the length of any unidirectional cycle.

STEP 0. The offsets between target (x', y') and current (x, y) positions are computed.

$(dx, dy) := (x' - x, y' - y)$.

STEP 1. The deviations are contracted to the interval $[-M/2, +M/2]$.

$dx := \bar{dx}$ IF $|dx| > M/2$; $dy := \bar{dy}$ IF $|dy| > M/2$

If $sign(dx) = sign(dy)$ then the minimal path is already determined and the diagonal is used as one of the subpaths. Note that the path length is given by $max(|dx|, |dy|)$ if the signs are equal, by $|dx| + |dy|$ otherwise.

STEP 2. One of the following operations is performed, only if $dx \cdot dy < 0$. They comprise a test whether the path with or without using the diagonal is shorter.

$$dx := \bar{dx} \text{ IF } |dx| > |dy| \text{ AND } |\bar{dx}| < |dx| + |dy| \quad // \, |\bar{dx}| = \max(|dx|, |dy|)$$

$$dy := \bar{dy} \text{ IF } |dy| \geq |dx| \text{ AND } |\bar{dy}| < |dx| + |dy| \quad // |\bar{dy}| = \max(|dx|, |dy|)$$

STEP 3. This step forces the agents to move in the same direction if source and target lie opposite to each other, namely at distance $M/2$ on the same axis. Thereby collisions on a common node on inverse routes are avoided.

$(dx, dy) := (|dx|, |dy|)$ IF $(dx = -M/2)$ AND $(dy = -M/2)$
$dx := |dx|$ IF $(dx = -M/2)$ AND $(dy = 0)$
$dy := |dy|$ IF $(dy = -M/2)$ AND $(dx = 0)$

Then a minimal route is computed as follows:

(a) If $dx \cdot dy < 0$ then
$[dz' = 0]$ move FIRST $dx' = dx$ steps, THEN move $dy' = dy$ steps

(b) If $dx \cdot dy > 0$ then calculate
(1) $dz' = sign(dx) \cdot \min(|dx|, |dy|)$ // steps on the diagonal
(2) $dx' = dx - dz', \quad dy' = dy - dz'$

$[dy' = 0]$ move FIRST dz' THEN dx', or
$[dx' = 0]$ move FIRST dy' THEN dz' .

This algorithm yields a minimal route and uses a cyclic priority for the six directions, two or one of them which are used in a valid minimal route. For short, the algorithm uses the priority scheme:

$[dx' = 0]$ move FIRST dy' THEN dz',
$[dy' = 0]$ move FIRST dz' THEN dx',
$[dz' = 0]$ move FIRST dx' THEN dy'.

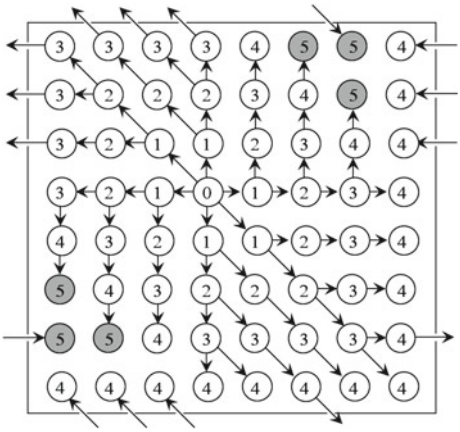


Fig. 6.7 This directed graph is a spanning tree of the torus showing the minimal path according to the XYZ–protocol from a source node "0" to any other node for a 8×8 network ($n = 3$, $N = 64$). The maximal distance (longest path) is the diameter $D_3 = 5$ for this graph (refer back to Eq. (6.1). Six antipodals are highlighted. Note that for n even ($n > 2$) this routing scheme would display twelve antipodals [50]

This priority scheme means: use "FIRST `dirR` THEN `dirL`" where `dirR` and `dirL` define the "right" minimal subpath and the "left" minimal subpath respectively,[1] viewed from the "observer" agent as in Fig. 6.6.

The minimal routes (FIRST `dirR` THEN `dirL`) are depicted as directed graph in Fig. 6.7 for a 8×8 network. The number in the nodes represents their distance from the source node "0".

6.2.4 Deterministic Routing

From the above protocol "FIRST `dirR` THEN `dirL`" the agent follows always the "right" minimal subpath. This means that the agent changes its moving direction accordingly. The minimal path can be computed only once at the beginning and stored in the agent's state. During the run, the agent updates the remaining path to its target, decrementing its `dirR` counter until zero, then decrementing its `dirL` counter if any, until completion. It is also possible to recompute the minimal path at each new position. This was done in the simulation and that yields the same result.

The problem with deterministic routing is that it is not deadlock-free (see deadlock discussion in Sect. 6.4.3). Another problem with this protocol is that it may not be optimal with respect to throughput, especially in case of congestion. But it should be noted that congestion usually is not very high, because there are *six* channels available in each node. Formally, the deterministic routing is secure for an agent alone.

[1] An equivalent symmetric protocol would be "FIRST `dirL` THEN `dirR`".

6.2.5 Adaptive Routing

The objective for adaptive routing is (i) to increase throughput, and (ii) to avoid or reduce the probability of deadlocks. This protocol was manually designed and it is a simple algorithm defining the new direction of an agent. During the run, if the temporary computed direction (e.g., `dirR`) points to an occupied channel, then the other channel (e.g., `dirL`) is selected no matter this channel is free or not. A minimal adaptive routing may be roughly denoted as "EITHER `dirR` OR `dirL`". The path from source to target remains minimal on the condition that it remains inside the boundaries defined by the minimal parallelogram.

In order to increase the throughput when the system is congested or to avoid deadlocks securely, the agent's behavior could be more elaborated. The agent could obey to an internal control automaton (finite state machine as in [39]) and this automaton can be optimized by genetic algorithms [51].

It would also be useful to allow the agent to *deviate* from the minimal route in case of congestion. Referring back to Fig. 6.6, going from source A to target B, the agent could route *out* of its minimal parallelogram and move within an extended area like the trapezium AP_1BR_3 or even the rectangle AP_1BR_1 although the minimal route is of course prioritized. As a consequence, *three* possible moving directions, instead of *two*, remain adaptively possible. The three other directions backwards are not allowed.

6.3 Modeling the Multi-Agent System

This section is the core of this chapter. The dynamics of moving agents is described, the impact of the *copy–delete* rules in the CA–w and CA models is emphasized and some programming issues are revealed.

6.3.1 Dynamics of the Multi-Agent System

The node structure, the channel state and how agents and arbiters cooperate are presented herein, the priority rule derived from the above adaptive protocol is analyzed and the conflict-free transition moving the agents follows.

6.3.1.1 Node Structure

The whole system consists of $2^n \times 2^n$ nodes arranged as in the T-grid of Fig. 6.5. Each node labeled by its (x, y) coordinates contains the 6–fold set

$$\mathcal{C} = \{C_0, C_1, C_2, C_3, C_4, C_5\} = \{E, SE, S, W, NW, N\} \tag{6.3}$$

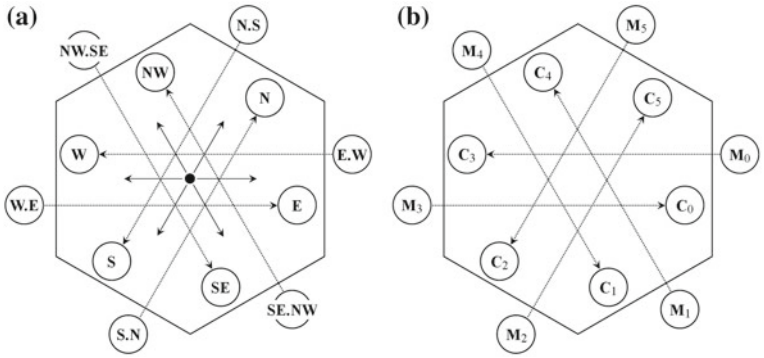


Fig. 6.8 Inner channels of $\mathcal{C}$ oriented (**a**) and labeled clockwise (**b**). Outer channels of $\mathcal{M}$ located in adjacent nodes. The direct neighbor of channel C_i in the adjacent node is denoted by M_i. The cardinal notation "$W.E$" stands for the E–channel of the W–neighbor; the same relative neighborhood is valid for any pair (node, channel) by symmetry: $M_{j\equiv i+3 \pmod 6}$ denotes the i–channel of the adjacent j–neighbor

of channels C_i oriented[2] and labeled clockwise (Fig. 6.8). Index i is called *position* or *lane number* in this context. The position of a channel defines also an implicit direction that defines the next *adjacent* node that an agent visits next on its travel. The direct neighbor of channel C_i in the adjacent node is denoted by M_i where

$$\mathcal{M} = \{M_0, M_1, M_2, M_3, M_4, M_5\} \tag{6.4}$$

and $M_{j\equiv i+3 \pmod 6}$ denotes the i–channel of the adjacent j–neighbor by symmetry. In the cardinal notation, e.g. for $i = 0$, "$W.E$" stands for the E–channel of the W–neighbor.

6.3.1.2 Channel State

Each agent has a direction which is updated when it moves. In other words, the current direction of the agent defines the channel in the next node where the agent requests to move to.

The i–channel's *state* at time t is defined by

$$c_i(t) = (p, (x', y')) \tag{6.5}$$

where (x', y') stands for the agent's target coordinates and $p \in \mathcal{P}$ stands for the agent's direction (a pointer to the next channel) in the set

$$\mathcal{P} = \{-1, 0, 1, 2, 3, 4, 5\} \equiv (\texttt{Empty}, \texttt{toE}, \texttt{toNW}, \texttt{toS}, \texttt{toW}, \texttt{toSE}, \texttt{toN}) \tag{6.6}$$

[2] Except a homeomorphism.

including an empty channel encoded by $\omega = -1$. In a graphical representation, the directions can be symbolized by $(\rightarrow\nwarrow\downarrow\leftarrow\searrow\uparrow)$ according to the inset in Fig. 6.5.

6.3.1.3 Agents and Arbiters

According to the above *adaptive* routing scheme, an agent can move to a *3–fold* subset of channels at most. Recall that the three other directions backwards are not allowed. For example, coming from channel $W.E$ of W–neighbor at $(x-1, y)$, going to channel E or N or SE of current node (x, y) as shown in Fig. 6.9a. In the same way, agents located in outer channels $NW.SE$ and $S.N$ are possible competitors for a part of this subset $\{E, N, SE\}$: channel subset $\{SE, E, S\}$ can be requested by un agent in $NW.SE$ while channel subset $\{N, NW, E\}$ by un agent in $S.N$. The *intersection* of those three requested subsets is the channel E. From this observation, E can be chosen as *arbiter* of three possible concurrent agents. In other words, a priority rule can be locally defined for *this* channel. Arbiter E is C_0 in Fig. 6.9b and this concurrent scheme is invariant by rotation.

This interaction between requesting agents and arbiter channels is formalized hereafter. Let the 3–uple of channels

$$\mathcal{C}_i = (C_{i+1}, C_i, C_{i-1}) \tag{6.7}$$

and let us denote by

$$\mathcal{M}_i = (R_i, S_i, L_i) \tag{6.8}$$

the ordered 3–uple opposite to C_i and where

$$R_i = M_{i+4}, \quad S_i = M_{i+3}, \quad L_i = M_{i+2} \tag{6.9}$$

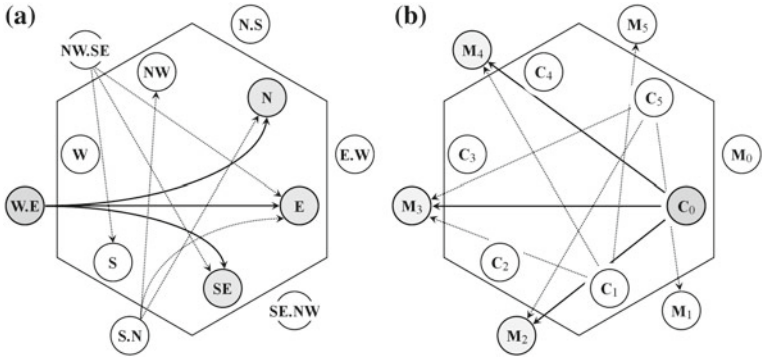


Fig. 6.9 **a** An agent located in the E–channel of the western node $W.E$ can move to one channel in the "opposite" subset $\{E, N, SE\}$. Two agents in channels $NW.SE$ and $S.N$ are possible competitors for a part of this subset. **b** As a consequence, channel C_0 is the arbiter of three possible concurrent agents in the requesting channels M_2, M_3, M_4

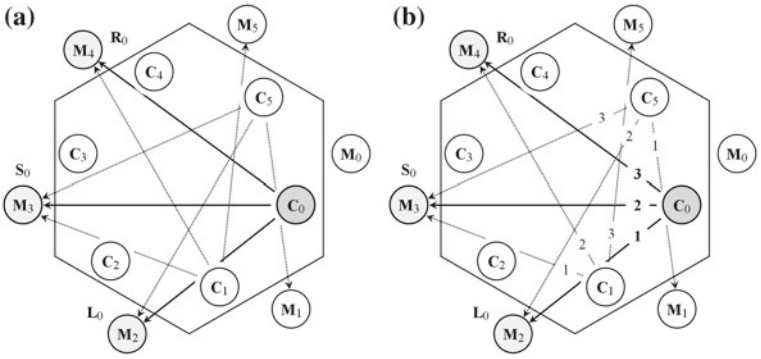


Fig. 6.10 **a** Requesting channels R_0, S_0, L_0 with respect to C_0. The "`right`", "`straight`", "`left`" respective directions are viewed from the "observer" C_0. The same concurrent scheme is valid all around from rotational invariance. **b** Priority rule: a priority order is assigned clockwise by any channel $C_i \in \mathcal{C}$ to its own requesting ordered set $\mathcal{M}_i = (R_i, S_i, L_i)$: `1` to `left`, `2` to `straight`, `3` to `right`

are the `right`, `straight` and `left` outer channels for the three possible incoming agents viewed from the "observer" channel C_i in Fig. 6.10a ($i = 0$ assumed herein). This 3–uple is of special interest because, viewed from channel C_i, the agents may only move from R_i or S_i or L_i to C_i on their route.

Conversely, $\mathcal{C}_i$ is the requested channel subset for S_i, as well as $\mathcal{C}_{i-1}$ for L_i and $\mathcal{C}_{i+1}$ for R_i. Now

$$\mathcal{C}_{i-1} \cap \mathcal{C}_i \cap \mathcal{C}_{i+1} = \{C_i\} \tag{6.10}$$

from (6.7). This simple but important property allows to define a *local* priority rule for channel C_i and invariant by rotation.

6.3.1.4 Priority Rule

Each channel $C_i \in \mathcal{C}$ computes the three exclusive conditions selecting the incoming agent that will be hosted next, with a priority assigned clockwise (Fig. 6.10b):

1. Agent wants to move from L_i to C_i, priority 1: $\texttt{LtoC} = (l = i)$
2. Agent wants to move from S_i to C_i, priority 2: $\texttt{StoC} = (s = i) \wedge \neg\texttt{LtoC}$
3. Agent wants to move from R_i to C_i, priority 3: $\texttt{RtoC} = (r = i) \wedge \neg\texttt{StoC}$.

In other words, this rule selects a winner among the three possible concurrent agents requesting channel C_i and the selection is assigned clockwise: first to `left`, second to `straight`, third to `right`, orientation viewed from the observer *channel*. It should be pointed out that this priority scheme is consistent with the protocol "FIRST `dirR` THEN `dirL`" defined in Sect. 6.2.3 but now viewed from the observer *agent*.

6.3.1.5 Moving the Agents

The above priority scheme ensures a *conflict-free* dynamics of moving agents[3] in the whole network. Thus, from the three previous conditions in channel C_i, five cases are distinguished:

- case $\kappa_1 : (p \neq \omega)$ // channel not empty, agent stays at rest,
- case $\kappa_2 : (p = \omega) \wedge \texttt{LtoC}$ // channel empty, agent to be copied from L_i,
- case $\kappa_3 : (p = \omega) \wedge \texttt{StoC}$ // channel empty, agent to be copied from S_i,
- case $\kappa_4 : (p = \omega) \wedge \texttt{RtoC}$ // channel empty, agent to be copied from R_i,
- case $\kappa_5 : (p = \omega) \wedge \neg\texttt{LtoC} \wedge \neg\texttt{StoC} \wedge \neg\texttt{RtoC}$ // channel remains empty.

The new target coordinates $(x', y')^*$ in the channel's state are either invariant if the agent stays at rest (case κ_1) or are copied from L_i or S_i or R_i exclusively, depending of the selected incoming agent hosted and to be received by the channel. Thus the target coordinates $(x', y')^*$ are updated as

$$\begin{aligned}
(x', y')^* &= (x', y') \quad \text{IF } \kappa_1 \\
(x', y')^* &= (x', y')_{L_i} \quad \text{IF } \kappa_2 \\
(x', y')^* &= (x', y')_{S_i} \quad \text{IF } \kappa_3 \\
(x', y')^* &= (x', y')_{R_i} \quad \text{IF } \kappa_4
\end{aligned} \tag{6.11}$$

according to the current channel's state and to the result of the selection.

Since the agent's target coordinates are stuck within its state, the agent must clearly carry them with it when moving. The new pointer to the next node

$$p^* = \varphi_{xy}\,((x', y')^*) \quad (p^* \in \mathcal{P}) \tag{6.12}$$

is then updated by φ_{xy} which yields the new direction from the target coordinates of the selected agent (φ_{xy} is the local updating function in the current node (x, y)). For case κ_5, the direction is irrelevant and the channel remains empty. Finally, the i–channel's *state* at time $t + 1$ becomes

$$c_i(t + 1) = (p^*, (x', y')^*) \tag{6.13}$$

and the new state is updated synchronously. It is assumed that the agents are initially placed on a channel which is part of the minimal route, and the initial direction is one of the minimal directions.

[3] Except special deadlock or livelock situations pointed out in Sect. 6.4.3.

6.3.2 *The CA–w and CA* Copy–Delete *Rules*

The synchronous transition (6.13) to the next timestep is governed by the *copy–delete* operating mode of either the CA or the CA–w model in Fig. 6.1. The impact of their rules is examined hereafter, that will highlight the simplicity induced by the *write*–access in the CA–w model.

6.3.2.1 CA–w Rule

The CA–w model is especially useful if there are no write conflicts by algorithmic design. This is here the case, because an agent is copied by *its* receiving channel, after applying the abovementioned priority scheme and according to the mode displayed in Fig. 6.1c. Thus only *this* receiving channel is enabled to delete the agent on the sending channel at the same time. A further advantage is that only the short-range *copy*–neighborhood is sufficient to move an agent, the wide-range *delete*–neighborhood (necessary for CA modeling and described hereafter) is not needed.

Therefore the 3–fold *copy*–neighborhood $\mathcal{M}_i$ that needs to be checked by C_i in order to receive the hosted agent is given by (6.8), this agent in $\mathcal{M}_i$ is released by C_i when firing the transition (6.13) and following the CA–w *delete–copy* operating mode in Fig. 6.1c.

6.3.2.2 CA Rule

The CA rule differs from the CA–w rule in the fact that the sending channel has to delete *itself* the agent when moving. This means that a separate *delete*–rule is necessary.

In order to release its own agent, the sending channel must be aware of the whole situation in its wide-range neighborhood defined as follows. Knowing that an agent in channel $M_{i+3} \in \mathcal{M}_i$ and wanting to move (Fig. 6.9a) will be selected by its arbiter which belongs to $\mathcal{C}_i$ (Fig. 6.9b), this requesting agent has other possible competing agents lying in $\mathcal{M}_{i+1}$ for C_{i+1} or $\mathcal{M}_i$ for C_i or $\mathcal{M}_{i-1}$ for C_{i-1}. Therefore, the full set of competitors is the union

$$\widehat{\mathcal{M}_i} = \mathcal{M}_{i+1} \cup \mathcal{M}_i \cup \mathcal{M}_{i-1} \tag{6.14}$$

but

$$S_{i-1} = L_i, \quad R_{i-1} = L_{i+1} = S_i, \quad S_{i+1} = R_i$$

from (6.9) whence

$$\widehat{\mathcal{M}_i} = (M_{i+1}, M_{i+2}, M_{i+3}, M_{i+4}, M_{i+5}) \tag{6.15}$$

or, if we exclude the own channel,

$$\hat{\mathcal{M}}_i = (M_{i+1}, M_{i+2}, M_{i+4}, M_{i+5}). \tag{6.16}$$

In addition, the sending channel must also be aware of the *move-to* conditions of C_i, whence the extended *delete*–neighborhood

$$\hat{\mathcal{M}}_i \cup C_i = (LL_i, L_i, R_i, RR_i) \cup (C_{i+1}, C_i, C_{i-1}) \tag{6.17}$$

with seven channels altogether and where "LL_i" and "RR_i" denote the wide-range left and right outer channels in Fig. 6.11b. As a matter of fact and referring back also to Fig. 6.10, it can be observed that only the "isolated" channels are excluded for this wide-range neighborhood.

Note that the cardinality of the neighborhood of a receiving channel is *three* whereas it is *seven* for a sending channel, without counting the own channel. Thus the whole neighborhood in the CA model is the union of the *copy*–and–*delete*–neighborhood with three and seven channels respectively, namely *ten* channels altogether when firing the transition (6.13) and following the CA *delete–copy* operating mode in Fig. 6.1a.

The discrepancy between both CA and CA–w rules described hereabove highlights henceforth the simplicity of the CA–w model. It should be noted that the channels of a node can be seen as the partitions of a cell as in "partitioned CA" [52]. Therefore a similar modeling can be done by partitioned CA. Another way of modeling such a system would be to use a hexavalent FHP–like lattice-gas [20]; but here the purpose is to avoid the two-stage timestep in order to save time, with only one clock cycle instead of two.

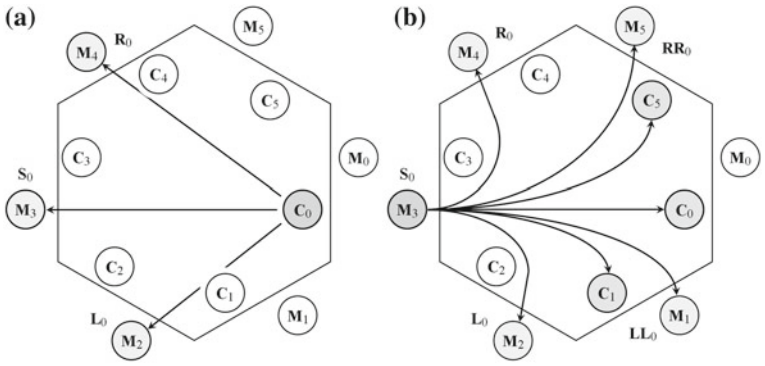


Fig. 6.11 Wide–range neighborhoods: **a** The 3–neighborhood $\mathcal{M}_0$ of receiver C_0. **b** The 7–neighborhood $\hat{\mathcal{M}}_0$ of sender S_0 extended to LL_0 and RR_0. Channels C_0 and S_0 coincide either as receiver or as sender

6.3.3 Programming Issues

Some more details are revealed for the reader interested in writing a simulation program. The following pseudo-codes show the algorithms for implementation of the CA and CA–w models. Missing items and notes "(*)" are explained afterwards.

6.3.3.1 CA Model

```
FOR EACH cell IN cellfield DO
  // compute own move condition, requires evaluation of copy- and delete neighborhood
1 cell.agent.can_move = (no agent or obstacle in front) AND                    (*)
                        (no other agent with higher priority can move to cell in front)

  // for each possible sending neighbor, requires evaluation of the copy-neighborhood
  // check if an agent wants to move to me and select one with the highest priority
2 neighbor(sending).agent.can_move(to_me) =
                              neighbor(sending).is(agent) AND
                              no other agent with higher priority wants to move to me

  // compute moving rule
3 cell_next<- empty   IF cell.is(empty) AND no agent can move to me // remains empty
           <- agent   IF cell.is(agent) AND receiving cell occupied // blocked
           <- empty   IF cell.is(agent) AND cell.agent.can_move     // delete (*)
           <- neighbor(sending).agent
                IF cell.is(empty) AND neighbor(sending).agent.can_move(to_me) // copy
ENDFOREACH

FOR EACH cell IN cellfield DO
5  cell <- cell_next // synchronous updating
ENDFOREACH
```

Statement 1 evaluates if the agent situated on its cell can move; this evaluation requires an extended neighborhood because of possible conflicts, namely the union of the copy- and delete neighborhood. Statement 2 evaluates if a neighboring agent can move to the current cell; this evaluation requires the copy-neighborhood only. In Statement 3 an agent may move, by deleting it by the sending cell and copying it by the receiving cell. Statement 5 performs a synchronous updating of the whole cell field.

The following simulation using the CA–w model is more simple.

6.3.3.2 CA–w Model

```
FOR EACH cell IN cellfield DO
  // for each possible sending neighbor, requires evaluation of the copy-neighborhood
  // check if an agent wants to move to me and select one with the highest priority
2 neighbor(sending).agent.can_move(to_me) =
                              neighbor(sending).is(agent) AND
                              no other agent with higher priority wants to move to me

  // compute moving rule, now without delete
3 cell_next<- empty   IF cell.is(empty) AND no agent can move to me // remains empty
```

```
            <- agent   IF cell.is(agent) AND receiving cell occupied // blocked
            <- neighbor(sending).agent
                  IF cell.is(empty) AND neighbor(sending).agent.can_move(to_me) // copy

  // extra CA--w operation: write on neighbor (deletion of agent when moving)
4 neighbor(sending).agent <- empty IF neighbor(sending).agent.can_move(to_me) //delete
ENDFOREACH

FOR EACH cell IN cellfield DO
5  cell <- cell_next // synchronous updating
ENDFOREACH
```

Compared to the CA program, the evaluation of the move condition (Statement 1 in CA program (*)) is omitted. Therefore the delete-neighborhood need not to be checked. As a consequence, the `delete(*)` line in Statement 3 of the CA program is also omitted and replaced by the additional Statement 4. Through this statement an agent on a sending cell is deleted by the receiving cell. The advantage of the CA–w program is that it is more concise and less expensive, because the `move` condition in Statement 1 needs not to be computed.

6.4 Router Efficiency and Deadlocks

Two test cases will be used for evaluation, where k is the number of agents, s the number of source nodes and m the number of target nodes:

1. **First Test Case** ($m = 1, k = s$). All agents move to the same common target. We will consider the case $k = N - 1$, meaning that initially an agent is placed on each site (except on the target). In this case the optimal performance of the network would be reached if the target consumes six messages in every timestep ($t = (N - 1)/6$). In addition, the target location is varying, with a maximum of N test configurations in order to check the routing scheme exhaustively. We recall that the T–grid is vertex-transitive, so the induced routing algorithm must yield the same result for all N cases!
2. **Second Test Case** ($k = s = m$). The sources are mutually exclusive (each source is used only once in a message set) as well as targets. Source locations may act as targets for other agents, too. We consider the case $k = N/2$ that was also used in preceding works [39, 40] for comparison. Note that the minimum number of timesteps t to fulfil the task is the longest distance between source and target which is contained in the message set. For a high initial density of agents the probability is high that the longest distance is close to the diameter of the network. Thus the best case would be $t = D_n$.

6.4.1 Efficiency of Deterministic Routing

Using one agent only in the router, it will travel always on a minimal route. More agents are also using a minimal route, but sometimes they have to wait due to traffic congestion.

6.4.1.1 First Test Case

In the first test case scenario, $k = N - 1$ messages move to the same common target from all other nodes. All possible or a large number of initial configurations differing in the target location were tested (Table 6.1). The results are the same for all tested initial configurations. This means that the router works totally symmetric as expected. An optimal router would consume in every generation six agents at the target, leading to an optimum of $t_{opt} = k/6$. It is difficult to reach the optimum, because the agents would need a global or a far view in order to let the agents move simultaneously in a cohort. Here an agent needs an empty receiver channel in front in order to move, thus empty channels are necessary to signal to the agents that they can move.

As the router is completely filled with agents at the beginning (one agent in each node except the target node), there exist some agents which have as travel distance the diameter D_n. Therefore the ratio t/D_n (B/C in Table 6.1) is significantly higher than one, slightly higher than $\sqrt{N}/2$. On the other hand, the ratio $t/(k/6) = B/D$ is quite good and relatively constant, that is $B/D \approx 2$ for large N, which is almost optimal because each agent needs an empty channel in front when moving without deviation on the minimal route. This phenomenon is easy to understand and has a close relationship with Traffic Rule 184 in a 1–dimensional system: a car with a car straight ahead cannot move and must wait for the next timestep.

Table 6.1 First test case: $k = N - 1$ messages travel from all disjoint sources to the same common target

Nodes N	Number of configurations (destinations) checked	(B) Message transfer time [steps]	(C) Diameter	Ratio B/C	(D) N/6	Ratio B/D
$4 = 2 \times 2$	all 4	1	1	1	1	1
$16 = 4 \times 4$	all 16	5	2	2.5	3	1.67
$64 = 8 \times 8$	all 64	23	5	4.6	11	2.09
$256 = 16 \times 16$	all 256	89	10	8.9	43	2.07
$1024 = 32 \times 32$	all 1024	351	21	16.7	171	2.05
$4096 = 64 \times 64$	64	1384	42	32.95	683	2.03

Message transfer time (in *timesteps*) in the T–grid, averaged over the number of checked initial configurations. The time is independent of the position of the target

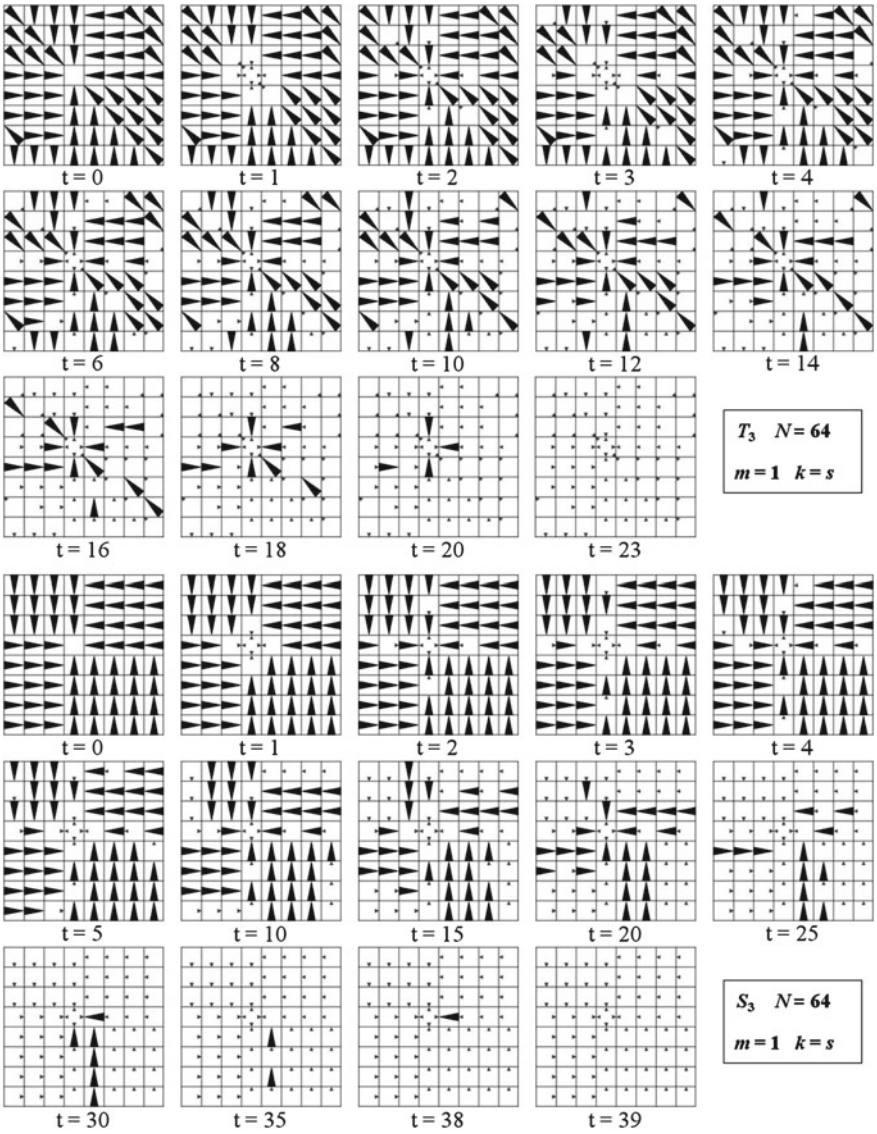


Fig. 6.12 Simulation snapshots for the first scenario in a 8×8 grid T_3, $N - 1$ agents moving to the same target position. Agents are depicted as black triangles, visited channels as small grey triangles: directions are symbolized by $(\rightarrow\nwarrow\downarrow\leftarrow\searrow\uparrow)$. Snapshots on S_3 are also displayed for comparison

A simulation sequence of this case is shown in Fig. 6.12 for the 8×8 grid T_3 and S_3 is also displayed for comparison.

6.4.1.2 Second Test Case

This test case was already used in a previous work [40] and is used for comparison. Therein, the agents were controlled by a finite state machine FSM: optimized, evolved agents were used, choosing a random direction with probability 0.3 % in order to avoid deadlocks, with only one agent per node. Ratio A/B in Table 6.2 shows that even the deterministic router with six channels performs significantly better, with $A/B \approx 2.5$ for $N = 1024$. The main reason is that here a node can host six agents, not only one, and therefore the congestion is considerably lower. The ratio (B/C) is noteworthy and shows that the mean transfer time is close to the diameter. This phenomenon is again easy to understand from Traffic Rule 184 but now in a fluid traffic. A simulation sequence of this case is shown in Fig. 6.13.

6.4.2 Efficiency of Adaptive Routing

An adaptive routing protocol was designed in order to speed up the message set transfer time and to avoid deadlocks during the run, although this could not be proved. When an agent computes a new direction and whenever the channel in that direction is occupied, the agent chooses the other minimal direction if there is a choice at all.

Table 6.2 Second test case: $k = N/2$ messages travel from disjoint sources to disjoint targets

Nodes N	Number of configurations checked for B randomly generated	(A) time steps, FSM controlled agent	(B) time steps, 6 channels non-adaptive	Ratio A/B	(C) Diameter	Ratio B/C	Time steps, 6 channels adaptive
$4 = 2 \times 2$	32	3.756	1	3.76	1	1	1
$16 = 4 \times 4$	256	8.528	2.520	3.38	2	1.260	2.520
$64 = 8 \times 8$	256	14.641	5.852	2.50	5	1.170	5.648
$256 = 16 \times 16$	256	28.848	12.070	2.39	10	1.207	11.574
$1024 = 32 \times 32$	256	58.438	23.367	2.50	21	1.113	22.648
$4096 = 64 \times 64$	256	128.087	45.199	2.83	42	1.076	44.082
$16384 = 128 \times 128$	128	300.330	87.789	3.42	85	1.033	86.668

Message transfer time (in *timesteps*) in the T–grid, averaged over the number of checked initial configurations. Routing with six channels per node performs significantly better (ratio A/B) than FSM controlled agents (one per node)

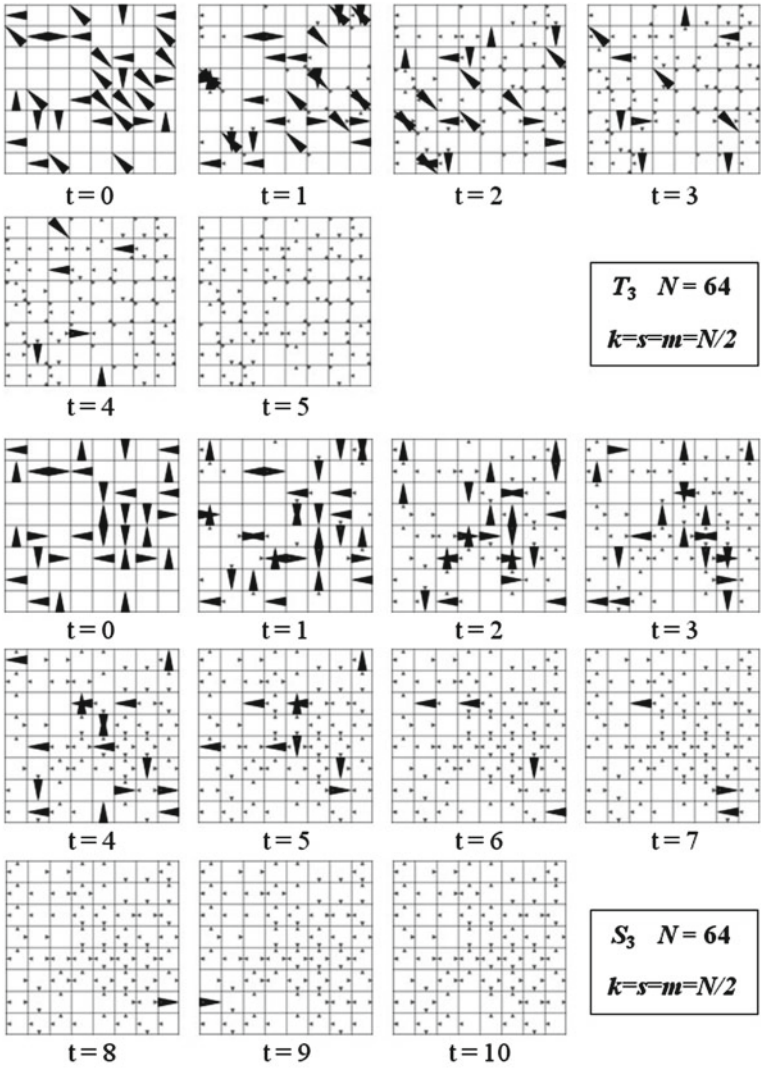


Fig. 6.13 Simulation snapshots for the second scenario in a 8×8 grid T_3, 32 agents moving to their assigned target. Agents are depicted as *black* triangles, visited channels as small *grey* triangles: directions are symbolized by $(\rightarrow\nwarrow\downarrow\leftarrow\searrow\uparrow)$. Snapshots on S_3 are also displayed for comparison

6.4.2.1 First Test Case

For this scenario with a common target the performance of adaptive routing is the same as for the deterministic routing. The reason is that all routes to the target are heavily congested. This means that the adaptive routing can hardly be better, but it is also not worse for the investigated case.

6.4.2.2 Second Test Case

For this scenario with randomly chosen sources and targets, the adaptive routing performs slightly—but only slightly—better. That means that agents' minimal path is seldom rerouted because of the fluid traffic. For example, for $N = 1024$, the message transfer time is reduced by 4.1 %. There seems to be more potential to optimize the behavior of the agents (using an FSM, or using a larger neighborhood) in order to guide them in a way that six agents are almost constantly consumed by the target.

6.4.3 Deadlocks

A trivial deadlock can be produced if all $6N$ channels contain agents (fully packed), thus no moving is possible at all. Another deadlock appears if $M = 2^n$ agents line up in a loop on all the channels belonging to one lane, and all of them have the same lane direction. Then the lane is completely full and the agents are stuck. To escape from such a deadlock would only be possible if the agents can deviate from the shortest path, e.g. by choosing a random direction from time to time. More interesting are the cyclic deadlocks where the agents form a loop and are blocking each other (no receiving channel is free in the loop). Two situations were investigated.

First situation `[Right Loop]` (Fig. 6.14). An empty node Ω is in the center of six surrounding nodes, let us call them A_0, A_1,... A_5 clockwise. Agent at A_0 wants to go to A_2, A_1 to A_3... in short the A_i want to go to A_{i+2} all around. Note that each agent has two alternatives: going first via Ω through the center or going first to a surrounding node (e.g., agent at A_0 can go first to Ω and then to A_2, or first to A_1 and then to A_2). Whether a deadlock appears depends on the initial assignments to the channels. If the initial assignments of all agents are "`use the left channel first`" via surrounding nodes, then the agents block each other cyclically. Otherwise they can move via the center node Ω and no deadlock occurs. Thereby it is assumed that the channels in Ω are empty or become empty after some time and are not part of other deadlocks.

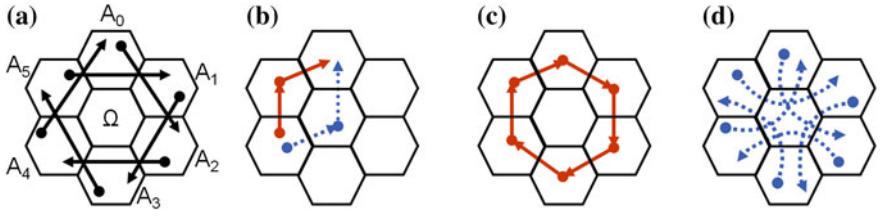


Fig. 6.14 A possible deadlock situation. **a** Agents targets, target = source + 2 mod 6. **b** The alternate paths, two min paths *solid* leftmost first, *dotted* rightmost first. **c** Cyclic deadlock appears if leftmost subpath is taken first. **d** No deadlock if rightmost subpath is taken first

Second situation `[Left Loop]`. This situation is symmetric to the right loop, except that the loop direction is now counterclockwise. A deadlock will appear if the initial directions of all agents are "`use the right channel first`".

If an initial configuration includes a right loop and a left loop, then at least one deadlock will appear if the initial assigned channel is fixed to the left or to the right. There are several ways to dissolve such deadlocks:

1. A spatial inhomogeneity is used, e.g., agents at *odd* nodes use initially the left subroute channel and agents at *even* nodes use the right subroute channel. The partition "odd–even" means $x + y \equiv 1$ or $x + y \equiv 0$ respectively, under addition modulo 2. This kind of partition, among others, was examined and did work for a limited set of experiments. Another similar way would be to randomize the initial subroute/channel assignment. This may be very successful but still there exists a very low probability for a deadlock.
2. The choice between the two minimal subroutes is taken randomly, or the choice may depends on an extended neighborhood.
3. It would be possible to deviate in a deterministic or non-deterministic way from the minimal route: for example an agent could move side-backwards if the whole area in direction of the target is blocked.
4. It would be possible to redistribute the channels during the run, by using a two-stage interaction-advection transition similar to the FHP lattice-gas [20]: move or don't move, then redistribute. In this case, the initially assigned channels and the used channels during the run could be dynamically rearranged.

6.5 Summary

The properties of a family of scalable 6–valent triangular tori were studied herein and for this family a minimal routing protocol was defined. A novel router with six channels per node was modeled as a multi-agent system within the cellular automata paradigm. In order to avoid the redundant computation of the moving condition, the CA–w model was used for implementation, that allows the receiving cell to copy the agent and to delete it on the sending cell. Thereby the description becomes more natural and the simulation faster. Both classical CA and new CA–w models were presented and compared. Each agent has a computed direction defining the new channel in the adjacent next node. The computed direction is a "minimal" direction leading on the shortest path to the target. The novel router is significantly faster (2.5 times for 1024 nodes) than an optimized reference router with one agent per node. In addition, an adaptive routing protocol was defined, which prefers the leftmost channel of a minimal route if the rightmost channel is occupied. Thereby a speed-up of 4.1 % for 1024 nodes was reached.

Deadlocks may appear for special situations when the system is overloaded, or when a group of agents form a loop. In order to avoid some of the deadlocks the initial subpath's direction can be alternated in space, or an adaptive protocol can be used.

The defined adaptive protocol switches to the other alternate minimal subpath in the case where the channel of the prior subpath is occupied. This adaptive protocol leads also to a higher throughput in the case of congestion. In order to dissolve deadlocks securely, a random or pseudo-random component should be introduced that may also allow the agents to bypass congested routes.

Further work can be aimed towards more intelligent agents in case of congestion through optimizing their behavior by using a genetic algorithm. Moreover, previous comparative works on the performance of agents moving either in the 4–valent S–grid or in the 6–valent T–grid [40, 53], with a speedup on T over S according to their diameter ratio (refer back to Eq. (6.2)), emphasize again our choice of triangular lattice explained in Sect. 6.1.3. The routing protocol could also be simplified by exploring the symmetries of the isotropic triangular grid: it is conjectured that this approach may drastically reduce the cost of the router, at least in a deterministic or adaptive context [54, 55].

References

1. Woolridge, M., Jenning, N.R.: Intelligent agents: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
2. Franklin, S., Graesser, A.: Is it an agent, or just a program?: a taxonomy for autonomous agents, In: Müller, J.P., Woolridge, M., Jennings, N.R. (eds.), Proceedings of ECAI'96 Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, pp. 21–35. Springer, New York (1997)
3. Holland, J.H.: Emergence: From Chaos To Order. Perseus Book, Cambridge (1998)
4. Woolridge, M.: An introduction to multiagent systems, 2nd Ed.. Wiley, New York (2002)
5. Pais, D.: Emergent collective behavior in multi-agent systems: an evolutionary perspective. Ph.D. Dissertation, Princeton University, Princeton (2012)
6. Schweitzer, F.: Brownian Agents and Active Particles. Collective Dynamics in the Natural and Social Sciences, Springer Series in Synergetics, Springer, Berlin (2003)
7. Shannon, CL. E.: Presentation of a maze-solving machine. In: 8th Conference of the Josiah Macy Jr. Found., Cybernetics pp. 173–180 (1951)
8. Blum, M., Sakoda, W.: On the capability of finite automata in 2 and 3 dimensional space. In: 18th IEEE Symposium on Foundations of Computer Science, pp. 147–161. (1977)
9. Fraigniaud, P., Ilcinkas, D., Guy, P., Andrzej, P., Peleg, D.: Graph exploration by a finite automaton. In: Fiala, J., et al. (eds.) MFCS 2004, LNCS 3153. pp. 451-462, (2004)
10. Spezzano, G., Talia, D.: The CARPET programming environment for solving scientific problems on parallel computers. Par. Dist. Comp. Practices **1**, 49–61 (1998)
11. Freiwald, U., Weimar, J.R.: JCASim—a Java System for Simulating Cellular Automata, Theory and Practical Issues on Cellular Automata, pp. 47–54. Springer, London (2001)
12. Rosenberg, A.L.: Cellular ANTomata. Adv Complex Syst. **15**(6) (2012)
13. Hoffmann, R.: Rotor-routing algorithms described by CA-w. Acta Phys. Polonica B Proc. Suppl. **5**(1), 53–68 (2012)
14. Hoffmann, R.: The GCA–w massively parallel model. In: Malyshkin, V. (ed.) LNCS 5698, pp. 194–206. (2009)
15. Hoffmann, R.: GCA-w: Global cellular automata with write-access. Acta Phys. Polonica B Proc. Suppl. **3**(2), 347–364 (2010)
16. Hoffmann, R.: GCA-w algorithms for traffic simulation. Acta Phys. Polonica B Proc. Suppl. **4**(2), 183–200 (2011)

17. Grünbaum, B., Shephard, G.C.: Tilings and Patterns. Freeman & Co., New York (1987)
18. Désérable, D.: A Terminology for 2*d* Grids, RR INRIA n° 2346, pp. 1–31 (1994)
19. Hardy, J., Pomeau, Y., de Pazzis, O.: Time evolution of a two-dimensional classical lattice system. Phys. Rev. Lett. **31**, 276–279 (1973)
20. Frisch, U., Hasslacher, B., Pomeau, Y.: Lattice-gas automata for the Navier-Stokes equation. Phys. Rev. Lett. **56**, 1505–1508 (1986)
21. Désérable, D., Dupont, P., Hellou, M., Kamali-Bernard, S.: Cellular automata in complex matter. Complex Syst. **20**(1), 67–91 (2011)
22. Morillo P., Fiol, M.A.: El diámetro de ciertos digrafos circulantes de triple paso. Stochastica **10**(3):233–249 (1986)
23. Bermond, J.C., Comellas, F., Hsu, D.F.: Distributed loop computer networks: a survey. J. Par. Dist. Comp. **24**:2–10 (1995)
24. Morillo, P., Comellas, F., Fiol, M.A.: Metric problems in triple loop graphs and digraphs associated to an hexagonal tessellation of the plane. TR 05–0286, pp. 1–6 (1986)
25. Davis, A.L., Robison, S.V.: The architecture of the FAIM-1 symbolic multiprocessing system. In: Proceedings of 9th International Joint Conferences on Artificial Intelligence pp. 32–38. (1985)
26. Davis, A.: Mayfly—a general-purpose, scalable, parallel processing architecture. J. LISP Symbolic Comput. **5**:7–47 (1992)
27. Chen, M.-S., Shin, K.G., Kandlur, D.D.: Addressing, routing and broadcasting in hexagonal mesh multiprocessors. IEEE Trans. Comp. **39**(1), 10–18 (1990)
28. Albader, B., Bose, B., Flahive, M.: Efficient communication algorithms in hexagonal mesh interconnection networks. J. LaTeX Class Files **6**(1):1–10 (2007)
29. Désérable, D.: Embedding Kadanoff's scaling picture into the triangular lattice. Acta Phys. Polonica B Proc. Suppl. **4**(2):249–265 (2011)
30. Gowrisankaran, C.: Broadcasting on recursively decomposable Cayley graphs. Discrete Appl. Math. **53**:171–182 (1994)
31. Désérable, D.: A family of Cayley graphs on the hexavalent grid. Discrete Appl. Math. **93**(2–3), 169–189 (1999)
32. Désérable, D.: Minimal routing in the triangular grid and in a family of related tori. In: Euro-Par'97 Par. Proceedings of Passau, Germany, LNCS 1300 pp. 218–225. (1997)
33. Désérable, D.: Broadcasting in the arrowhead torus. Comput. Artif. Intell. **16**(6), 545–559 (1997)
34. Heydemann, M.-C., Marlin, N., Pérennes, S.: Complete rotations in Cayley graphs. Eur. J. Comb. **22**(2), 179–196 (2001)
35. Fraigniaud, P., Lazard, E.: Methods and problems of communication in usual networks. Discrete Appl. Math. **53**(1–3), 79–133 (1994)
36. Dally, W.J., Seitz, C.L.: The torus routing chip. Distrib. Comput. **1**, 187–196 (1986)
37. Xiang, Y., Stewart, I.A.: Augmented k-ary n-cubes. Inf. Sci. **181**(1), 239–256 (2011)
38. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. Chem. Phys. **21**(6):1087–1092 (1953)
39. Ediger, P., Hoffmann, R., Désérable, D.: Routing in the triangular grid with evolved agents. J. Cell. Automata **7**(1), 47–65 (2012)
40. Ediger, P., Hoffmann, R., Désérable, D.: Rectangular vs triangular routing with evolved agents. J. Cell. Automata **8**(1–2), 73–89 (2013)
41. Loh, P.K.K., Prakash, E.C., Performance simulations of moving target search algorithms, Int. J. Comput. Games Technol. **2009**:1–6 (2009) (New York, Hindawi Publ. Corp.)
42. Korf, R.E.: Real-time heuristic search. Artif. Intell. **42**(2–3):189–211 (1990)
43. Ediger, P., Hoffmann, R., Routing based on evolved agents. In: 23rd PARS Workshop on Parallel Systems and Algorithms, ARCS, Hannover. Germany 2010, pp. 45–53 (2010)
44. Ediger, P., Hoffmann, R.: CA models for target searching agents. Automata São José dos Campos. Brazil, ENTCS **252**:41–54 (2009)
45. Shamei, A., Bose, B., Flahive, M.: Adaptive routing in hexagonal torus interconnection networks, IEEE High Performance Extreme Computing Conference (HPEC), (2013)

46. Leighton, F.T., Leiserson, C.E.: Wafer-scale integration of systolic arrays. IEEE Trans. Comput. **C-34**:448–461 (1985)
47. Cole, R.J., Maggs, B.M., Sitarman, R.K.: Reconfiguring arrays with faults—Part I: Worst-case faults. Siam J. Comp. **26**(6):1581-1611 (1997)
48. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks. Morgan Kaufmann, San Francisco (2002)
49. Hoffmann, R., Désérable, D.: Efficient minimal routing in the triangular grid with six channels. In: Malyshkin, V. (ed.) LNCS 6873 pp. 152–165, (2011)
50. Désérable, D.: Arrowhead and Diamond Diameters (unpublished)
51. Ediger, P.: Modellierung und Techniken zur Optimierung von Multiagentensystemen in Zellularen Automaten, Dissertation, TU Darmstadt, Darmstadt, Germany (2011)
52. Poupet, V.: Translating partioned cellular automata into classical type cellular automata. Journées Automates Cellulaires, Uzès, France pp. 130–140 (2008)
53. Hoffmann, R., Désérable, D.: All-to-All communication with cellular automata agents in 2d Grids—topologies, streets and performances. J. Supercomp. **6**9(1):70–80 (2014)
54. Désérable, D.: Hexagonal Bravais-Miller routing of shortest path (unpublished)
55. Miller, W.H.: A Treatise on Crystallography. J. & J.J, Deighton, Cambridge (1839)

Chapter 7
Multi-Resolution Hierarchical Motion Planner for Multi-Robot Systems on Spatiotemporal Cellular Automata

Fabio M. Marchese

Abstract This chapter presents a new multi-resolution and hierarchical approach to the problem of motion planning of Multi-Robot Systems on discretized spaces. The goal is to operate on large spaces (compared to the size of the robots), where the number of cells quickly becomes untreatable, in particular for n interacting robots problems, without losing precision (resolution). To work around this problem, we have introduced 3 levels of maps: the first is topological, the second a rectangular tessellation covering the free space, and the third a regular (small) cells decomposition. The first two maps are used to reduce the problem and to simplify it with non-accurate planning. Limiting the search space to smaller areas of interest at the last level and considering the interactions between robots, precise parallel motion planning is performed using Spatiotemporal Cellular Automata.

7.1 Introduction

This chapter concerns the coordinated motion of a Multi-Robot System (MRS) over wide spaces. We have considered MRSs composed of heterogeneous mobile robots having generic shapes, sizes and kinematics. In particular, we are interested in the coordination of a team of mobile robots moving over 2D structured environments (2D manifolds). Typical applications include the CMOMMT problem [18] related to surveillance tasks, as well as cooperation issues in soccer teams (e.g., RoboCup) or transportation of parts in manufacturing plants (logistic), rescue robots and equivalent problems. Over the years, various planners have been implemented on continuous spaces. Despite their interesting properties, they are prone to the problem of using high time consuming algorithm. We want to tackle an issue like extended spaces or, conversely, small models with very high-precision representation as an unique

F.M. Marchese (✉)
Università degli Studi di Milano-Bicocca, DISCo, V.le Sarca 336, Milan, Italy
e-mail: marchese@disco.unimib.it

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_7

problem. It is necessary to reduce calculation times throughout two solutions: (i) the adoption of models on discrete spaces (lattices or grids); (ii) the reduction of the complexity of the planning algorithm to reduce the search space. Starting from a 2D metric representation of the environment, in a completely automatic way, maps at different resolutions level are calculated, finally generating a topological space representation (graph of adjacent regions). The aim is to reduce the complexity of the problem on large lattices, in order to make them more manageable. Therefore, we realized a multi-resolution system, starting from a high-resolution map (regular cells decomposition). Then, rectangular regions of different sizes (clusters of cells) are recognized, generating an irregular decomposition or tessellation, but still able to completely cover the Free Space. It is similar to generalized cones by Brooks [4], where rectangles are considered as a degenerate case of cones. From this map, we generated a graph of adjacent rectangles and finally a graph of passing zones, which are used for a topological planning. Conceptually, we have designed a two-level hierarchical planner:

1. High level planner (Gross motion planning). This planner works on a topological map, the only purpose of which is to identify a 'channel' the robot moves through. This planning phase does not take into account the time level, but it works exclusively at the space level (geometric level, hence only trajectories). The channel delimits the area in which to look for a precise spatial trajectory (and later on a spatiotemporal trajectory, i.e. a movement).
2. Low level planner (Fine motion planning). It is a spatiotemporal CA (STCA) planner working on a precise environment map delimited by the above mentioned 'channel'. It coordinates the necessary maneuvers to avoid obstacles (in particular to avoid moving obstacles, i.e. other robots), while taking into account shapes, sizes, orientations, kinematics of robots and temporal constraints (departure time and arrival time).

The rest of the chapter is organized as follows: in Sect. 7.2, the MRS Motion Planning Problem is introduced. Sections 7.3 and 7.4 are related to Fine and Gross Motion Planning. Section 7.5 concerns to an example of MRS problem and in Sect. 7.6 the conclusions are presented.

7.2 MRS Motion Planning Problem

A Multi-Robot System is a set (a team) of robots sharing a common task. To solve the same task, the robots often use the same resources. In the same workspace, several MRS with different tasks can coexist, but they still compete for the same resources. In particular, while moving, they compete for the access to the 'Space' resource. In this work, we address the problem of the coordination of motion for a set of heterogeneous mobile robots in order to avoid collisions with static and dynamic obstacles. Since the late 70s, many approaches have been proposed for the solution of the Path/Motion Planning problem for single and multiple robots. A Configuration Space (C-Space)

solution has been proposed since 1979 [13] where a geometrical description of the environment was given. To address the problem in a dynamical world, some authors proposed the Artificial Potential Fields Methods as in [9]. Jahanbin and Fallside introduced a wave propagation algorithm in the C-Space on discrete maps (Distance Transform [8]). In the 90s, Barraquand et al. used the Numerical Potential Field Technique over the C-Space to build a generalized Voronoi Diagram [1]. In 1994, Zelinsky extended the Distance Transform to the Path Transform [22]. Since 1990, Warren in [21] used a discretized 3D C-Spacetime (2D Workspace + Time) for multiple translating robots' motion planning with simple shapes (only square and circle). In the same decade, the Cellular Automata approach for the single robot path-planning problem was introduced [2, 14, 20]. In [11], the authors applied the concepts of Game Theory and multiobjective optimization to the centralized and decoupled multi-robots motion-planning. A solution in the C-Spacetime has been proposed in [3], where the authors use a decoupled and prioritized path-planning in which they repeatedly reorder the robots' priorities to try to find a solution. In the 2000s, Cellular Automata started to be also used for the coordination of multiple robots [7, 16].

7.3 Fine Motion Planner

The Fine Motion Planner calculates the precise collision-free trajectories and movements that a team of robots must follow to reach their goals. Although it is driven by the Gross Motion Planner, it is the most important phase because it finalizes the method. It relies on a set of discretized metric spaces (see 7.3.1), the most important of which is the cellular space STCA. This planner uses the approach of Artificial Potential Fields, i.e. force fields that drive the robots toward their given goals.

7.3.1 Spaces

In this section, the spaces used by the fine motion planner are described. They are multidimensional Cellular Spaces (up to 5 dimensions) and all of them are projections of the underlaying discretized version of a 2D manifold in the real Space where the robots moves. All these spaces participate, by means of the evolutions of their automata, to the computation of the planner. In particular, the Attraction Space finalized the contribution of the other spaces, generating the final motion of the MRS.

7.3.1.1 Spatiotemporal Cellular Automata

All the spaces used derive from the definition of discretized spacetime. It is a multidimensional cellular space, i.e. a discretized version of the continuous linear

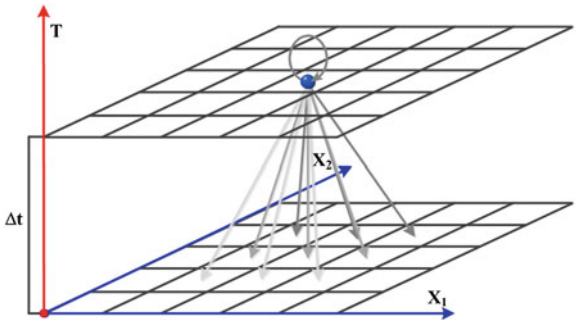


Fig. 7.1 Spatiotemporal cellular automata: dependencies between CA in adjacent layers and itself (radius 1)

(Euclidean) Spacetime, where some spatial dimensions are compounded with the Time's Arrow. Each dimension is discretized in indivisible quanta that represent the lower limit of both spatial and temporal resolution. This discretization induces a partition of multidimensional space in basic cells. Depending on the type of spacetime represented, each cell contains a value/state (even a vector of states) that is calculated on the basis of homologous values in adjacent cells with which it interacts. It is therefore a Cellular Automata, composed of different temporal layers. Each layer contains the representation of the whole workspace at a certain moment. Its peculiarity lies in the mapping of physical Time, not to be confused with Computing Time and Simulation Time. While physical Time is represented as a dimension of Spacetime, and therefore it becomes a static component, the Computation Time is the time necessary to upgrade the cells of STCA, and finally the Simulation Time is the passage from one temporal layer to the next of the STCA to follow the evolution of the system (the robots' movements). Since, as is well known, Time flows in one direction, to satisfy this property and then the physical feasibility, the Cellular Automata in a layer depend exclusively on Cellular Automata of the layer beneath (even more than one layer) at the same location or in the spatial neighborhood of the cell. It depends also on its own state, but it does not depend on automata of the same temporal layer or on those at successive times. Thus the number of total dependencies is reduced and, accordingly, computation time. For example, in a 3D Spacetime, the total number of dependencies from any other cell (neighborhood radius 1) would be 27, but thanks to the direction of Time, they are only 9 + 1 (Fig. 7.1). This condition is necessary to ensure the feasibility of the system, in both cases if it represents a real or a pure mathematical transformation.

7.3.1.2 C-Spacetime Space

When a layer of the STCA represents the configurations of a physical system, it becomes a discretized C-Spacetime. Please note that a configuration is a set of parameters representing the state of a system (in other words, the robot is represented as a point in the space). In our case, the configurations are given by the poses of

each robot. Because robots are considered as rigid bodies, a limited number of parameters represents their states: their positions and orientations. For motions on 2D spatial manifolds, we use (X, Y) for the position + Θ for orientation to define the Configuration Space. The overall C-Spacetime will be 4D: 2D (position) + 1D (orientation) + 1D (time). In the C-Spacetime, obstacles are mapped as not admissible configurations, that is configurations the robot cannot assume at a given instant of time without colliding with them (but which can be assumed in other moments if the obstacles move away).

7.3.1.3 Coordination Space

The Coordination Space is a unique space for all the MRSs, where the interaction robot-robot and robot-obstacle are modeled. It is the space in which the robots are coordinated in their movements, i.e. where to monitor the interactions between various objects (static and dynamic) to prevent their movements that lead them to collide. It is a representation of the real Spacetime in a discretized form, to manage movements on a 2D manifold. It is a 3D space: 2D (position) + 1D (time). In this space all static obstacles (e.g., walls) and dynamic obstacles are mapped, including moving objects (with a priori known movement) and all robots. All their movements are computed by the planner, in a similar way as in the discretized C-Spacetime, but the robots and all the other objects are fully represented, time by time, with their real shapes and extensions. It replaces what was once known as Repulsive Space, where artificial forces where evaluated to keep the robot away from the obstacles. In the Coordination Space, the cells are either occupied or free depending on whether an object at a certain moment covers that position. In order to satisfy Shannon's Theorem, Time is sampled at double frequency, i.e. the time unit used is half of the time unit used in any other spaces to prevent thin objects from passing through each other.

7.3.1.4 Attraction Space

The Attraction Space is a 5D STCA, where the configurations of all robots are represented. It is ultimately a 4D C-Spacetime for each robot, developed along the Robot Dimension, i.e. in each layer the C-Spacetime of a single robot is evaluated. Each automaton has a non-stationary potential value (it evolves along the Computational Time, but it is stationary in its representation in the Spacetime). This potential is referred to a lower potential located in the cell of the goal configuration. As usual, whenever there is a potential field, a force F appears ($-grad\phi$). In this metaphor, this virtual force attracts the robot towards the goal, turning around the obstacles. A potential value represents the integer 'distance' of the current cell from the goal cell along the shortest collision-free path (more simply: it is the minimum cost to reach the goal from the current cell avoiding all obstacles). The potential is updated with respect to the neighbors' values to reach a steady value. The updating rule is quite

simple: the potential value is the minimum value of the potentials of the neighbors plus the cost of the move that carries the robot from the neighbor cell to the current one. Within a temporal layer, the potentials are calculated up to their stability, before the successive layer is updated on the base of the current one (the temporal layers are sequentially updated). At the end, all temporal layers are stabilized and the 4D space contains a 'potential bowl' with a minimum at the goal pose. The CA updating rules ensure that the potential is calculated by turning around obstacles, and with a single minimum, i.e. no local minima are generated (in which the robot would stall). Within the Attraction Space all the movements are represented which may lead to the goal pose from any starting pose, under the form of spacetime trajectories (geometrization of movements). The calculation of the movements (ST trajectories) is made according to the negated gradient of the potential surface. The robot is shrunk to a single point, which moves on (rolls over) a potential 4D hyper-surface towards the goal. In order to take into account the actual size of the robot (which also changes with its orientation), it relies on the Coordination Space to validate the moves set, temporarily deactivating the moves which would lead to a collision (see 7.3.2.2).

7.3.2 Motion and Moves

Planning is the results of the interaction between two aspects: on one hand the search spaces (7.3.1), on the other hand the model of the process. In this section, it is introduced the kinematic model of a generic robot. A new 'dynamical' model of the robot results from combining the robot's kinematic model with its shape, called *Motion Silhouette*.

7.3.2.1 Spatiotemporal Moves

Robot discrete kinematics is described by a set of atomic spacetime moves, also including (non-)holonomic constraints. All objects, including static objects, follow a temporal trajectory (movement) consisting of a sequence of spacetime moves. In a discretized spatiotemporal space Z^4 for a robot moving on a 2D manifold, the definition of spatiotemporal move is the 4-tuple: $(\Delta x, \Delta y, \Delta\theta, \Delta t)$, where Δ is a finite variation, $(\Delta x, \Delta y) \in Z^2$, $\Delta\theta \in S^1$, $\Delta t \in Z$ and with the obvious constraint $\Delta t \geq 0$. This definition has two main interpretations: $(\Delta x, \Delta y, \Delta\theta)$ are finite increments of the spatial coordinates during the finite time interval Δt.

It entails the following space metrics Δs:

$$move \cong \frac{(\Delta x, \Delta y, \Delta\theta)}{\Delta t} \quad \Rightarrow \quad \Delta s^2 = \Delta x^2 + \Delta y^2 + r^2\Delta\theta^2$$

In the same way, $(\Delta x, \Delta y, \Delta\theta, \Delta t)$ are finite increments of the spatiotemporal coordinates, inducing the following metrics:

$$move \cong (\Delta x, \Delta y, \Delta\theta, \Delta t) \quad \Rightarrow \quad \Delta S^2 = \Delta x^2 + \Delta y^2 + r^2\Delta\theta^2 + v_r^2\Delta t^2$$

where ΔS is the 'distance' between two events of the spacetime, r is a dimensional constant, v_r is the 'speed' of spontaneous translation along the Time axis.

Static objects only have a single type of motion parallel to the Time's Arrow. They have only one move (we call it 'existence rule'): (0, 0, 0, +1). This move is part of the kinematics of all robots, without which the object does not have a physical sense (it should move indefinitely, something like a *perpetuum mobile*, never standing in a place).

Thanks to this definition of spatiotemporal move, the movement of a rigid body becomes a geometrical trajectory in the spacetime executed by means of a sequence of finite moves. The speed is computed as usual, but it is a rational value. For example, (+2, 0, 0, +1) is a move with double speed (x direction) or (+1, 0, 0, +2) represents a move at a half speed with respect to normalized units. Using an appropriate set of moves, it is possible to represent every type of kinematics of any robot. Furthermore, there is no theoretical limit to represent any velocity.

7.3.2.2 Motion Silhouette (Moves Validation)

To face with real robots in real environments, it is necessary to represent the shape of a robot. In the Configuration Space, the robot is represented only by a point (a vector of coordinates), but its real extension is not considered. In the late 70s, Lozano-Pérez first introduced the concept of obstacles enlargement to take into account the actual footprint of a robot [13]. In the first method, enlargement was isotropic, by an amount equal to the maximum radius of the robot (Fig. 7.2a). This method clearly led to an excessive enlargement because it transformed the robot shape to a cylinder that surrounded it. The result was a great loss of available free space and the closure of narrow passages. At the beginning of the 80s, Lozano-Pérez et al. [12] proposed a better solution: they also took into account robot orientation (Fig. 7.2b), realizing n obstacles maps, one for each expected orientation and with a different obstacles enlargement (anisotropic enlargement). This was translated in the C-Space as zones with admissible configurations and areas with forbidden configurations.

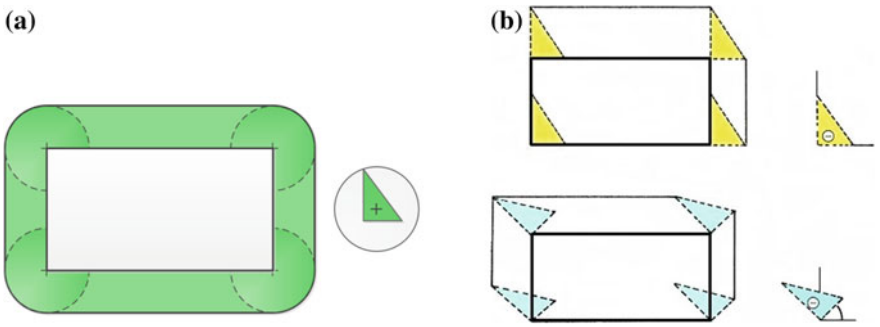


Fig. 7.2 Obstacle enlargement: **a** isotropic [12]; **b** oriented [13]

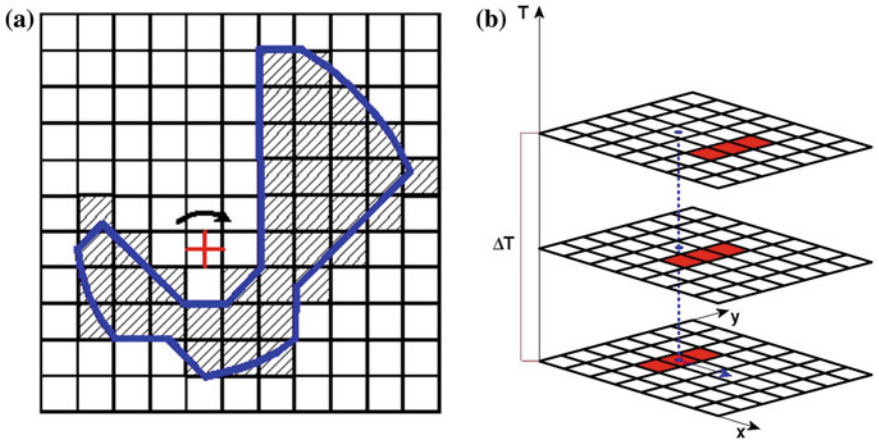


Fig. 7.3 **a** Sweeping silhouette [15]; **b** Motion silhouette [17]

In 2003, the authors proposed a method which takes into account what happen between two different orientations: the Sweeping Silhouettes [15]. Instead of widening the obstacles, the Sweeping Silhouettes of the robot were used on discretized C-Space cells to determine which configuration have to be considered admissible. This method makes it easier and more accurate to determine prohibited configurations, eliminating even the configurations in which the robot does not collide with obstacles just in the start and arrival poses of a single move, but also in all the intermediate positions, avoiding problems of 'tunneling' thin object like in Fig. 7.3a. The next evolution occurred in 2011 with the Motion Silhouette [17]. It coincides with the use of discretized C-Spacetime, strictly necessary when planning in the presence of moving obstacles (in particular other robots). With a double sampling of Time (to satisfy the Shannon's Theorem) a sequence of silhouettes are generated to map an elementary move. A collision test occurs within temporal layers of the Spacetime with respect to the poses of other obstacles. The prediction of a collision in a temporal movement step allows to disable the move starting from that particular robot pose (Fig. 7.3b). The test is performed for each pose and for each instant of time to validate the set of available moves (i.e. kinematics), temporarily disabling those which lead to a collision. Motion Silhouette method is able to represent any robot shape of any size. The only limit is due to the limit of the representation. It is a discretization of a 2D shape, thus the accuracy of the representation depends on the real size of a cell.

7.3.3 Interaction Between Spaces and Moves (Planning)

Conceptually, the motion planning is the result of the interaction between the Spaces and the set of Moves (kinematics) of a robot, i.e. depending on the move selected, spaces are altered accordingly.

7.3.3.1 Prioritized Planning

While the path planning problem for a single robot has a polynomial complexity, in 1979 Reif [19] established that the problem for a team of robots is PSPACE-hard, which implies it is NP-hard. Canny later established that the problem lies in the PSPACE, and therefore the general motion planning problem is PSPACE-complete [5]. Even Warehouseman's problem on a 2D grid is PSPACE-hard [6]. In general, for a N rigid robots problem the number of dimensions of the C-Spacetime would be: $C_{ST} = C^1 \times C^2 \times ... \times C^N \times T$. If robots are moving on a 2D manifold, the C-Space of a single robot is $\Re^2 \times SO(2) \equiv \Re^2 \times S^1$, thus the overall C-Spacetime has $3N + 1$ dimensions. Even for small MRS, the cardinality of the space makes the problem untreatable. Therefore, it is necessary to reduce the number of dimensions, for example adopting the Prioritized Planning technique (a description in [10]). This technique is a case of Decoupled Planning for multiple robots, where the robot to robot interaction is ignored in the first phase of motion design. Then the interactions are taken into account to constrain the available options. The problem arises when no option remains because this approach is not reversible, thus losing the completeness. Nevertheless, Prioritized Planning is very practical and solves most of the situations. In the prioritized approach, a planning order is given, starting to plan with the high-priority robot first. Robots with a lower priority see the higher-priority robots as moving obstacles with designed spatiotemporal trajectories. The planning phases are:

1. Establish a priority order for the robots (ordered set).
2. Plan the motion for the robot with the highest priority not yet planned (single robot motion planning in its Attraction Space).
3. Using the Coordination Space, select one collision-free movement from the set of all movements found (if any).
4. If there is at least one movement, trace the robot in the Coordination Space (mark the configurations along the movement as no longer available: the robot becomes an obstacle for all the other robots with a lower priority).
5. Exit if all the robot has been planned (finding or not a movement).
6. Goto step 2.

If no collision-free movement can be found for a robot, many recovery strategies can be adopted. The simplest one is to eliminate the robot from the space (and from the problem). Another strategy is to consider the robot as a static object standing in the starting pose. A more complex strategy is to let the robot at the starting pose, with the task to stay in the same pose, but reducing its priority. In this case, the robot can move away if another robot (with a higher priority) has to pass in that pose, and then it goes back to the initial pose. It is quite hard to define the complexity of this algorithm. It has been established [14] that, for a single robot, in the worst cases the complexity is $O(N^2)$, where N is the number of cells of the discretized C-Space. Hence, for an algorithm using the Prioritized Planning, its complexity is $O(pN^2)$, where p is the number of robots of the MRS.

7.3.3.2 T-Invariant Motion Planning

Dual kinematics is defined as a set of dual moves. These are obtained by the original kinematics by reversing the sign of all the components of the initial moves:

$move \cong (\Delta x, \Delta y, \Delta\theta, \Delta t) \quad dualmove \cong (-\Delta x, -\Delta y, -\Delta\theta, -\Delta t)$

The dual moves still belong to the same set of the original moves, and the dual kinematics is again a 'normal' kinematics. The original problem was a classical top-down planning (or backward planning: from goal to the start); the dual problem is a bottom-up planning (or forward planning: from start to goal). By exchanging the start with the goal and dualizing the kinematics we can solve the dual problem using the same algorithm. Hence, because they have the same solution, they belong to the same class of motion planning problems. It is possible to plan a movement from the goal to the starting pose, or viceversa from the starting pose to the goal, and we will find the same solution. We call this property as T-invariance of Motion Planning.

7.4 Gross Motion Planning

The approach to multidimensional Cellular Automata suffers from the problem of the appreciable increase of the computational burden, which grows with the number of dimensions of the Spacetime and the number of robots. Let's suppose we model a space with 100×100 cells, with a discretization of the orientation every 3 degrees (120 cells along the orientation axis), with 50 ticks along the time axis (100 cells for double sampling) and finally using 10 robots. The overall Attraction Space easily exceeds the billion of cells needed to manage the MRS (the Coordination Space is limited to only 10^6 cells). Fortunately, the Attraction Spaces of each robot are handled separately, but anyway each one reaches 10^8 cells. It is important to reduce its complexity. To do this, we add a preprocessing phase: the gross motion planning. Its purpose is to identify 'coarse' channels within which the robot will move. Then simply generate a set of possible trajectories (inside the channel) not taking into consideration the travel times. The aim is to discard all regions in the environment through which the robots do not pass and limit the precise motion planning only to the 'touched' areas, thus significantly reducing the number of cells really involved.

7.4.1 Topological Approach

To drastically reduce the computation time, we decided to adopt a static topological approach (not considering the time at this phase). From the 2D grid map of the environment, through appropriate processing, we generated a weighted graph representation of space (topological map). Topological map building occurs in three

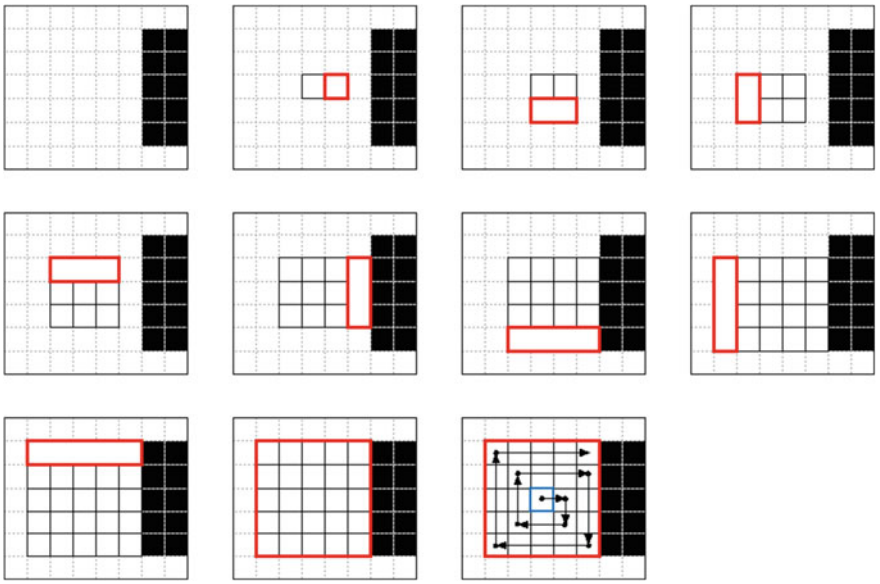


Fig. 7.4 Example of clockwise spiral cells clustering

steps:

1. free space decomposition in rectangular regions (complete coverage) through a mechanism of aggregation of cells;
2. search for border areas between regions;
3. construction of topological weighted graph.

7.4.1.1 Cells Clustering (Rectangular Coverage)

The process of decomposition of the free space in rectangles is performed by clustering adjacent cells in a spiral pattern (Fig. 7.4). It identifies a cell seed not yet part of any rectangle. This cell represents a basic 'rectangle' from which to start an enlargement phase by aggregation. It proceeds with a clockwise spiral by adding a row of cells (a rectangle of one cell thick) to the previous rectangle, getting a new wider rectangle. The result is always a rectangle, i.e. a convex region (the rectangles set is closed under this operation). The procedure continues while it finds new lines of cells that do not overlap with obstacles, or with the map boundary, or with other rectangles already identified, or until it reaches a maximum width.

At this point, it tries with a new cell not yet used and repeats the clustering. The procedure terminates when all cells are part of a rectangle (red in Fig. 7.5a) having the complete coverage of the space.

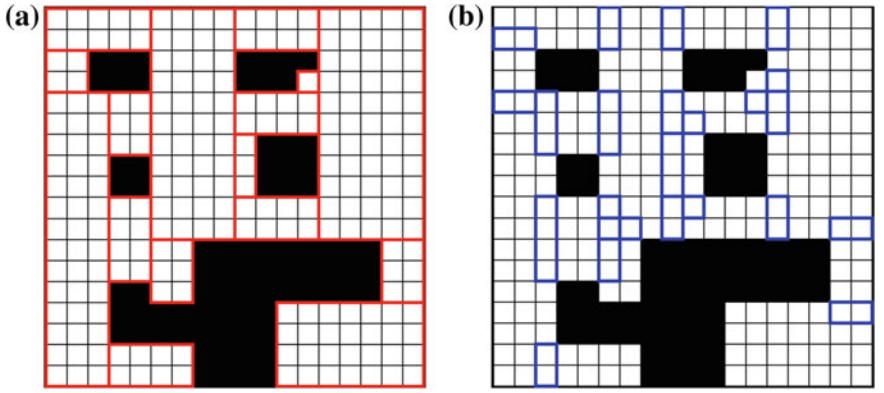


Fig. 7.5 Cells clustering: **a** *Rectangular* tessellation; **b** *Border rectangles* ('doors')

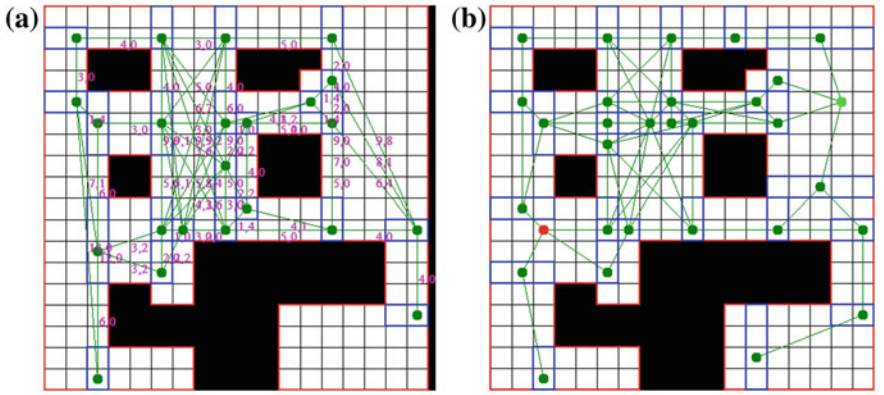


Fig. 7.6 Topological map: **a** weighted graph; **b** insertion of starting and goal nodes

7.4.1.2 Calculation of Boundary Zones

From the set of rectangular regions, border areas between them are determined. The algorithm considers pairs of rectangles to determine if there are groups of border cells between them. These cells will constitute a new set of rectangles one cell thick (blue in Fig. 7.5b). During the movement, a robot that has to come into an area has to go through one of these thin rectangles ('doors' or 'passages'), representing the input door-sill of the rectangles. If they are wider than robot size, they will become the nodes of a graph (Fig. 7.6a), while links will represent a chance to switch between different sills crossing the rectangle in which they are contained. As part of border regions of convex areas, links cannot pass over occupied cells, ensuring the clearance. The links are weighted by values representing the linear distance between the sills, according to the canonical Euclidean metric. This graph will be used for the Gross Motion Planning.

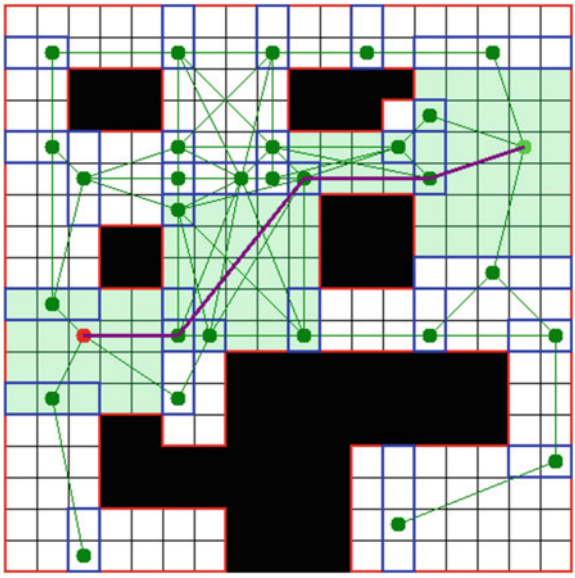


Fig. 7.7 Topological path ('channel' in light green)

7.4.1.3 Topological Path-Planning

Planning starts with the inclusion of two additional nodes that represent the starting point (green) and arrival point (red in Fig. 7.6b). These nodes are connected to all the existing door-sills of the rectangle, altering the topology (the algorithm cuts any other links inside the rectangle because it will never participate in the final trajectory. Likewise for the group of nodes connected to the arrival rectangle).

Topological level planning is done using the classic Djikstra's algorithm on weighted graphs (with exclusively positive weights), identifying the lowest cost path. This cost is only a rough estimate of the true length of the trajectory, and the path obtained is not optimal from many points of view. The purpose of this phase is just to identify the rooms involved by the trajectory in order to find a passing channel (Fig. 7.7), not to compute the path to be followed. With subsequent computation of a precise movement, it can determine the true trajectory inside the channel. Since the fine planner computation is relatively heavier, if we restrict its application solely to space involved by the channel, we can significantly reduce the computation times in some cases even by a factor of 10.

7.4.2 Examples

In this section, we show some examples related to the topological phase (or gross motion planning).

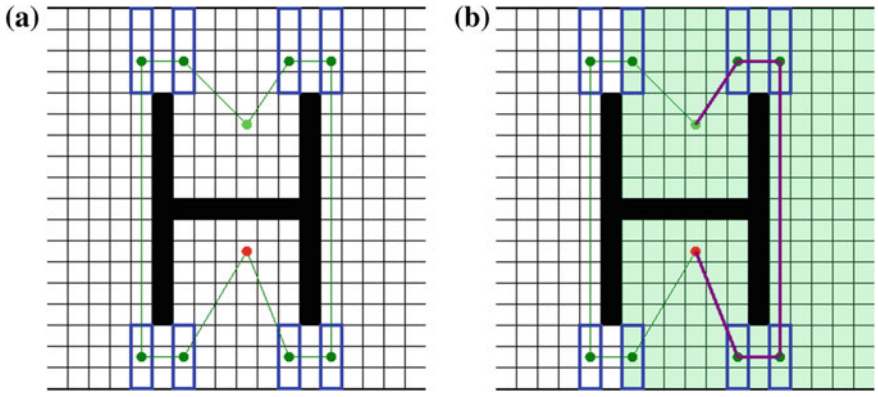


Fig. 7.8 H-shaped obstacle problem: **a** graph; **b** 'channel'

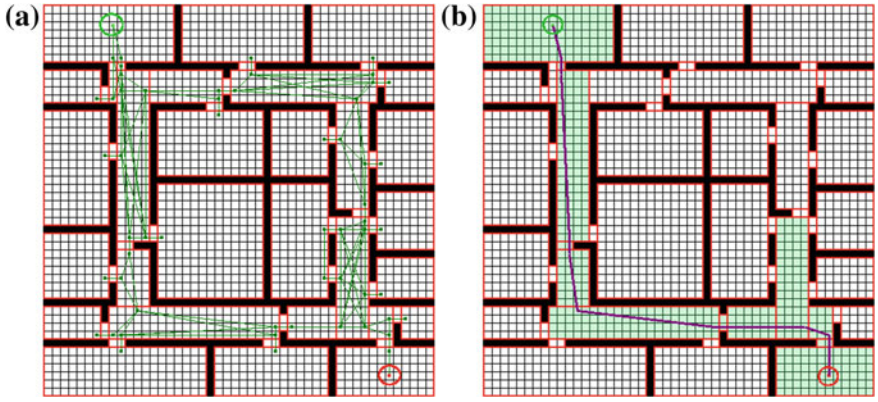


Fig. 7.9 Office-like environment: **a** graph; **b** 'channel'

7.4.2.1 H-Shaped Obstacle

Figure 7.8 shows that using nodes at the 'doors' of the rectangular regions, since they are convex, the planner does not stall in concave obstacles, but it always drives the robot to exit easily from a potentially critical situation.

7.4.2.2 Office-Like Environment

In the situation of Fig. 7.9, an office-like environment is represented. The total number of cells is 2,500 (50 × 50 cells). During the fine motion planning, the total number of cells of the Attractive Spacetime rises to 18 Mcells (18×10^6 cells). With the topological planning phase, the number of cells is reduced to about 3.5 Mcells, i.e. less than 20 %.

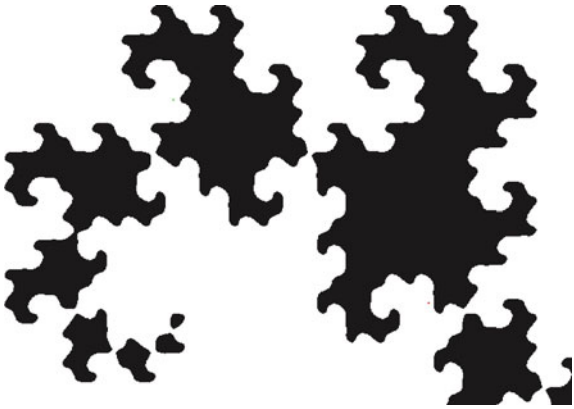


Fig. 7.10 Fractal obstacles example

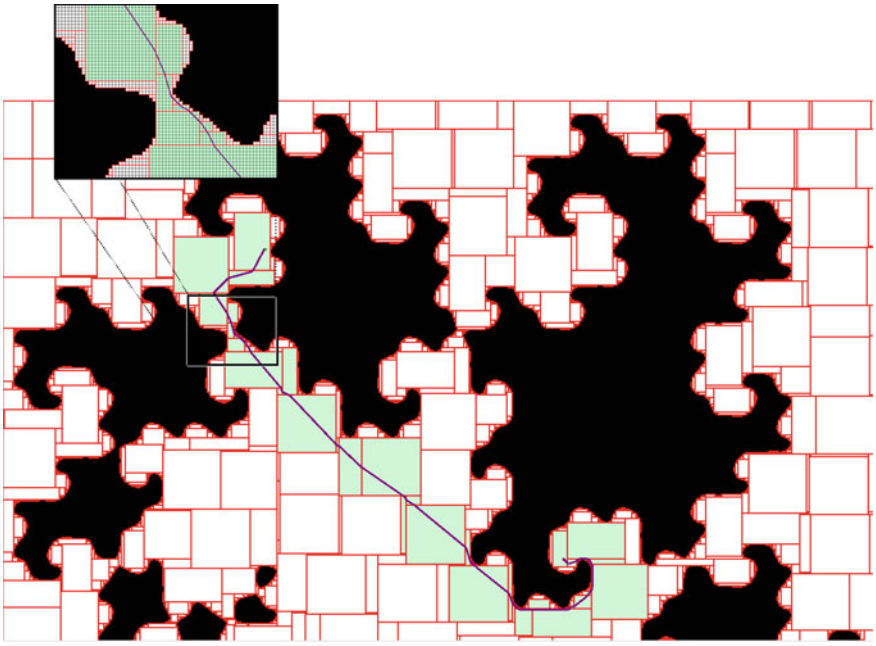


Fig. 7.11 Fractal obstacles channel

7.4.2.3 Fractal Obstacles

The last two examples were set to highly stress the system. In Fig. 7.10 a binarized fractal is represented on a grid of 650 × 450 cells (292,500 cells). In this situation, the Attractive Spacetime for a single robot would be more than 2.1 billions cells!.

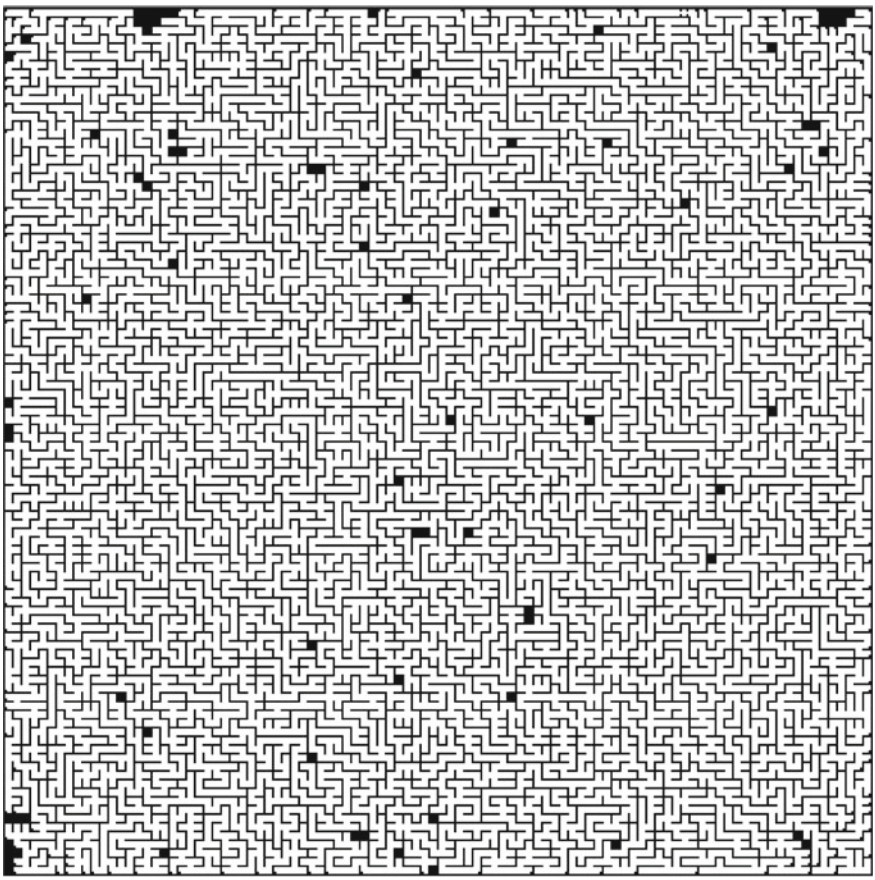


Fig. 7.12 The Maze

Making a topological planning and limiting the rectangles size to 10 % of the length of the environment sides (45 cells), we will get a total of 1,967 rectangles (Fig. 7.11). The trajectory obtained involves only 19,371 cells out of 292,500 (6.6 %), reducing the total amount of cells involved in the next phase up to 139.5 Mcells. Some computation times: even in such a hard situation, the most costly phase (graph building) requires only 0.15 s (on a Intel 32bit 8core 3.6 GHz), while the planning phase just 0.12 s.

7.4.2.4 Huge Maze

This example represents the most complex situation, although it is not realistic for a robot. A maze of 400 × 400 cells (160,000 cells) has been built (Fig. 7.12) using an automatic algorithm, making sure to generate at least 3 cell wide passages. It is

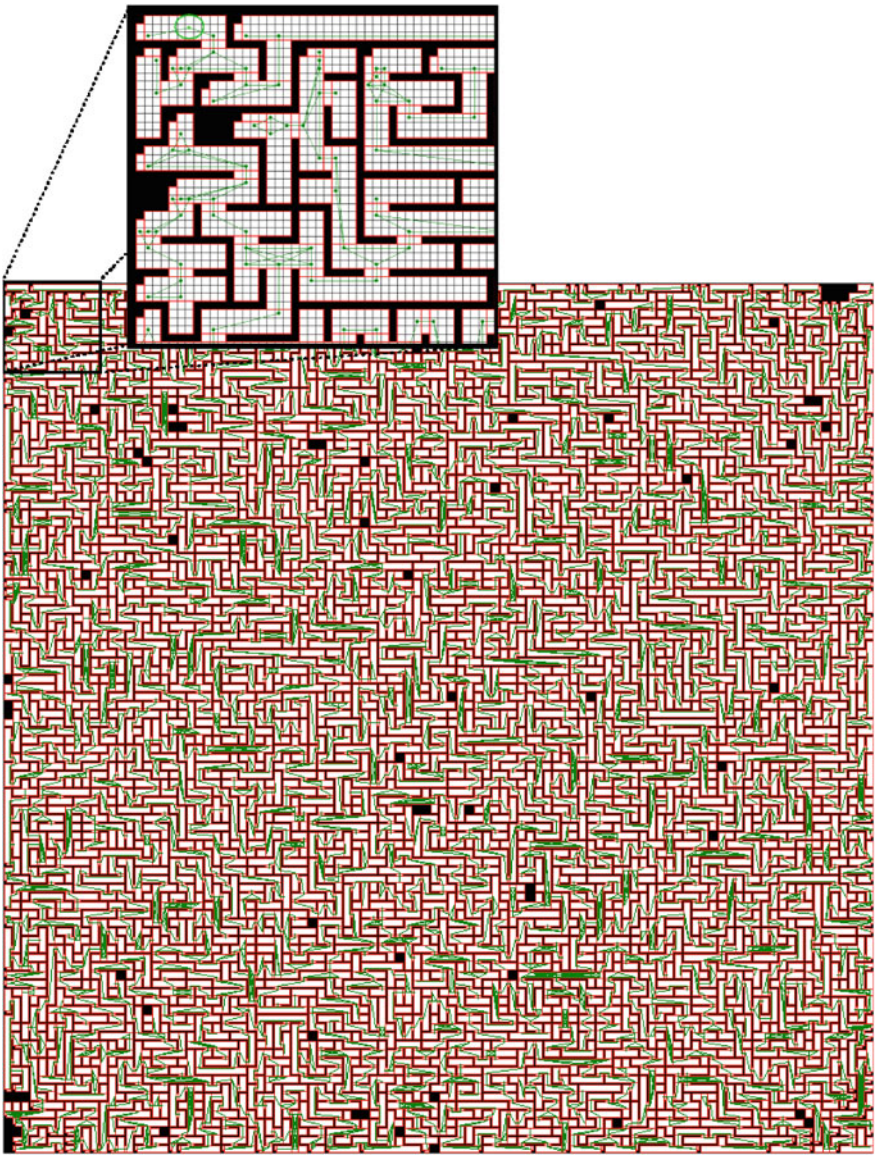


Fig. 7.13 Maze rectangular tessellation

unrealistic for a robot because it would be wide at most one cell to ensure the passage, but it represents a very hard test for this algorithm.

In Fig. 7.13 the set of rectangular regions is shown. The average computation time (on 1,000 repetitions) for this phase is 0.87 s.

Two problems have been tested: in the first the robot starts from the top-left corner and has to reach the bottom-right corner. In the second test a path covering a wider

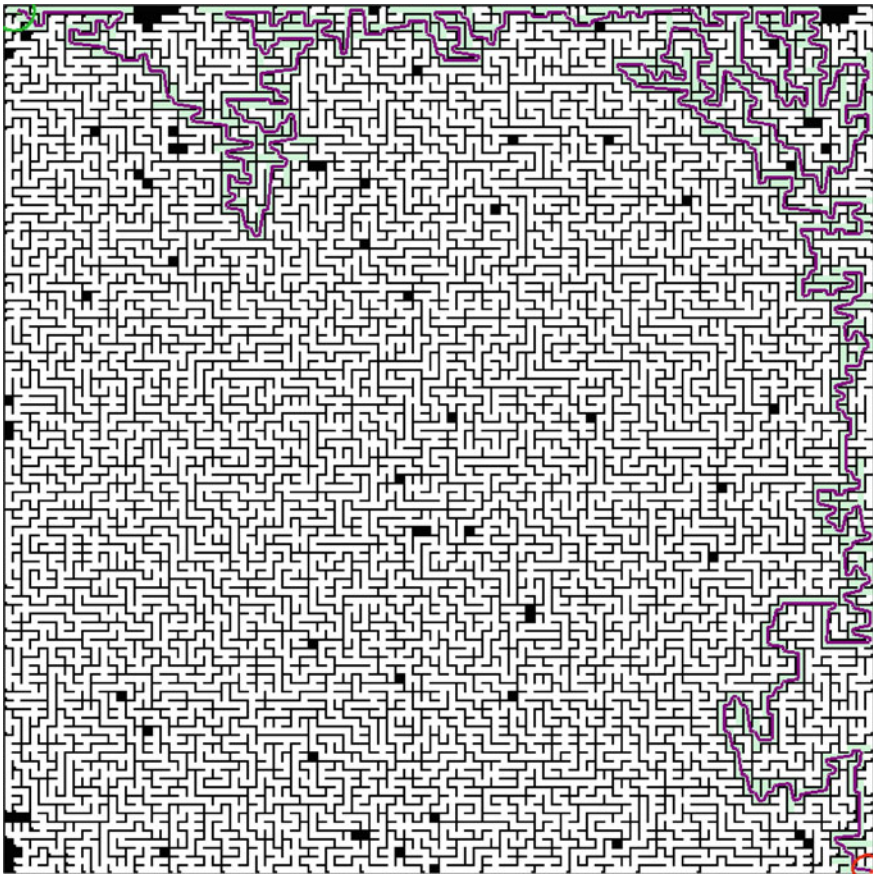


Fig. 7.14 Maze: first problem solution

area is found. The first result is shown in Fig. 7.14. The average time to solve the problem is 0.29 s.

The second problem is shown in Fig. 7.15 and requires 0.39 s. If we use the fine motion planner on this maze, we would have a total number of cells in the Attractive Spacetime of 4.6 Gcells for a single robot. With the topological phase to reduce the involved space, we would need only respectively 7 % of cells for the first issue and 25 % for the second (which deliberately has a broader coverage).

7.5 Multi-Robots Motion Problem

In a situation with multi-robots, the gross motion planning allows to decouple the problem moving a single robot at a time. The adjacency graph (relatively heavier at the computational level) remains unchanged and is calculated off-line once for all the robots: from time to time the goal and start nodes of a robot are added to calculate its

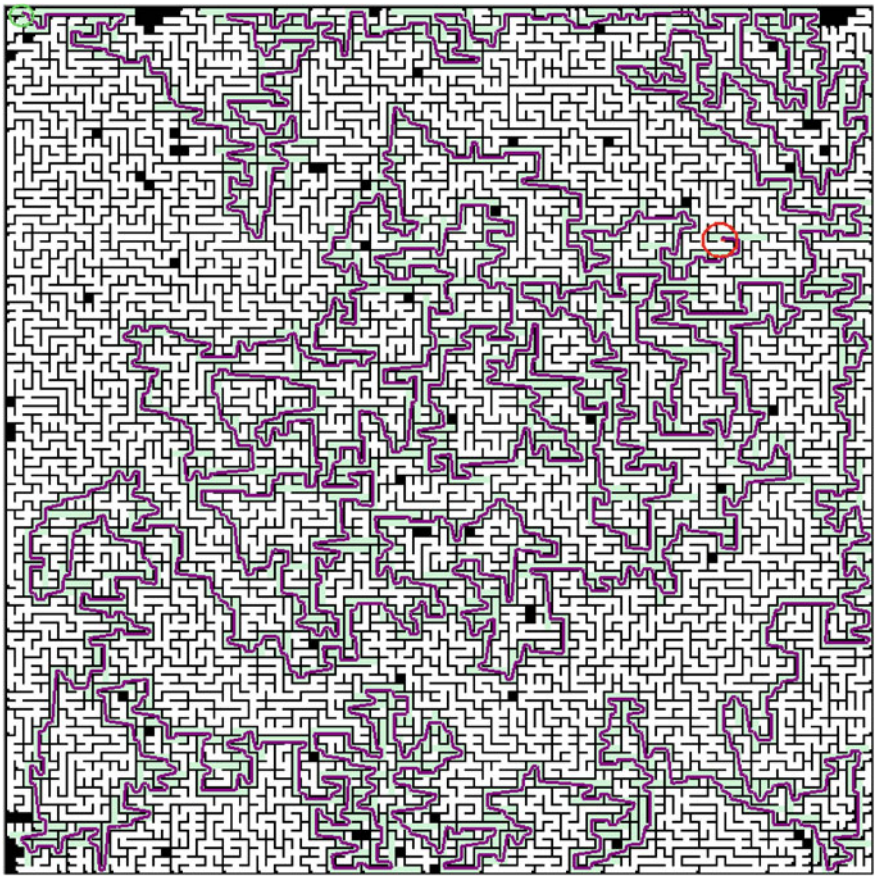


Fig. 7.15 Maze: second problem solution (path with a wider coverage)

trajectory (or to determine that it does not exist). This decoupling enables to locate the coarse paths for all the robots, but also has a second important utility: to determine the areas of interaction (interference) between robots. Identifying which robots pass through the same room, it figures out which robots risk mutual collision and which are excluded. It realizes a decoupling of the problem of multiple robots: on one hand there are robots that move through the environment without any risk of collisions and require just a planner for a single robot (polynomial complexity) applied separately (in parallel and non-interacting). On the other hand, there will be groups of robots that risk reciprocal collisions and that will be processed by a coordinated motion planner for MRS (coordinated within each group, but in parallel between different groups). In all cases, the level of complexity significantly decreases on the basis of the number of robots in addition to the space dimensions. If, as in the example of Fig. 7.16, instead of using a coordinated planner over four robots on the entire space, we use one on only two robots and a single planner on the remaining two, besides

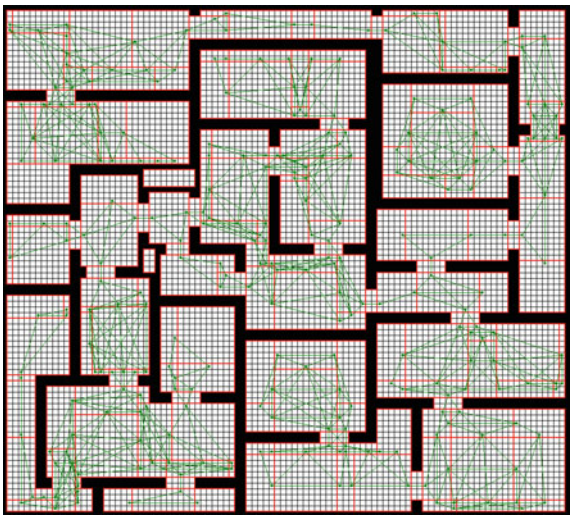


Fig. 7.16 Multi-robots motion problem graph

Table 7.1 Multi-robots motion problem performances (topological phase)

Phase	Mean time (ms)
Graph build	0.639
Green R. planning	0.046
Red R. planning	0.045
Blue R. planning	0.031
Yellow R. planning	0.030

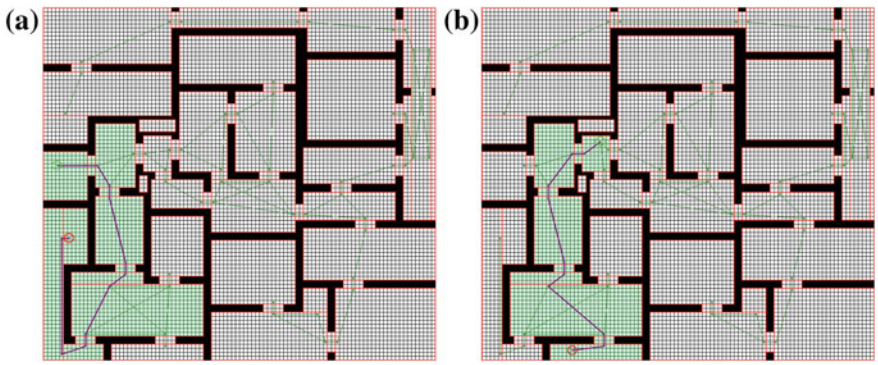


Fig. 7.17 *Green robot* (**a**) and *Red robot* (**b**) topological paths

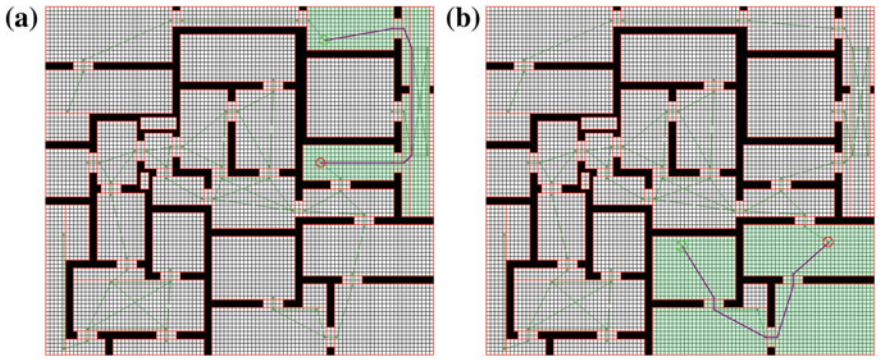


Fig. 7.18 *Blue robot* (**a**) and *Yellow robot* (**b**) topological paths

Table 7.2 Multi-robots motion problem performances (Spatiotemporal planning phase)

Phase	Mean time (s)
Green+Red R. planning	0.83
Blue R. planning	0.50
Yellow R. planning	0.84

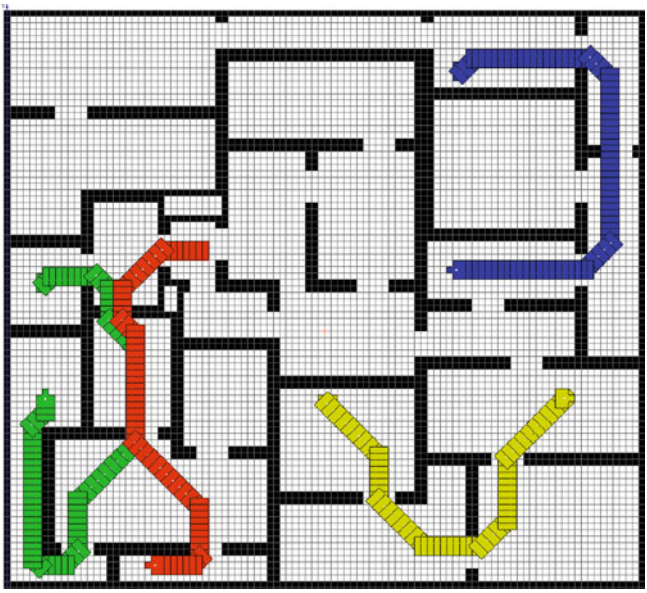


Fig. 7.19 Summary of the precise trajectories of the *Green*, *Red*, *Blue* and *Yellow robots*

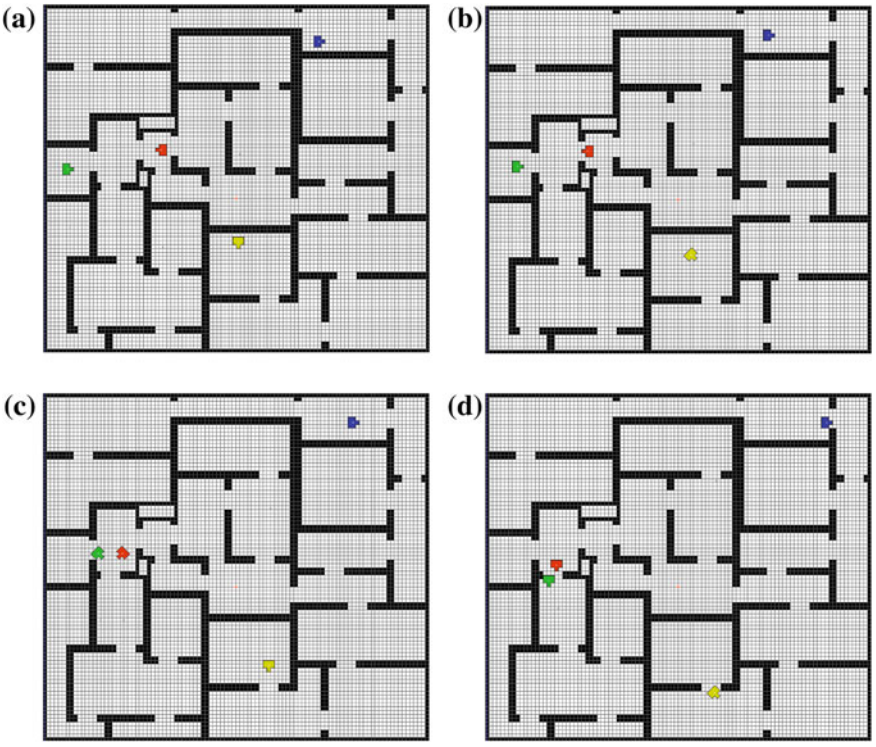


Fig. 7.20 MRS movement (1): **a** T = 0 (robots start moving); **b** T = 5; **c** T = 12 (the interaction between *Red* and *Green robots* starts); **d** T = 20 (the *Green robot* has higher priority and passes first)

reducing the space involved in the planning, the complexity is much lower. For this problem, the average calculation time is less than one millisecond for all phases, including the construction of the graph, as shown in Table 7.1 (Figs. 7.17, 7.18).

The calculation of the graph is made once for any problem data set with any number of robots and can be made off-line. The calculation for each robot takes place in real-time in parallel on different processors, and the worst result is for the Green robot with 0.046 ms, a time absolutely negligible. In the next step, we use the fine motion planner to determine the precise trajectories, maneuvering to handle the interaction between robots in the same rooms. The result is shown in Fig. 7.19, where the trajectories have been summarized. As we can see, the spatiotemporal motion planner finds slightly different but more efficient trajectories, handling the real robots shapes and their orientations; in particular, it manages the interaction between Red and Green robots to avoid collisions.

In Figs. 7.20, 7.21, snapshots at significant timestamps are reported. It is interesting to note the interaction phase between Red and Green robots starting at $T = 12$ to $T = 45$. Actually, the calculation has been assigned to three instances of the same

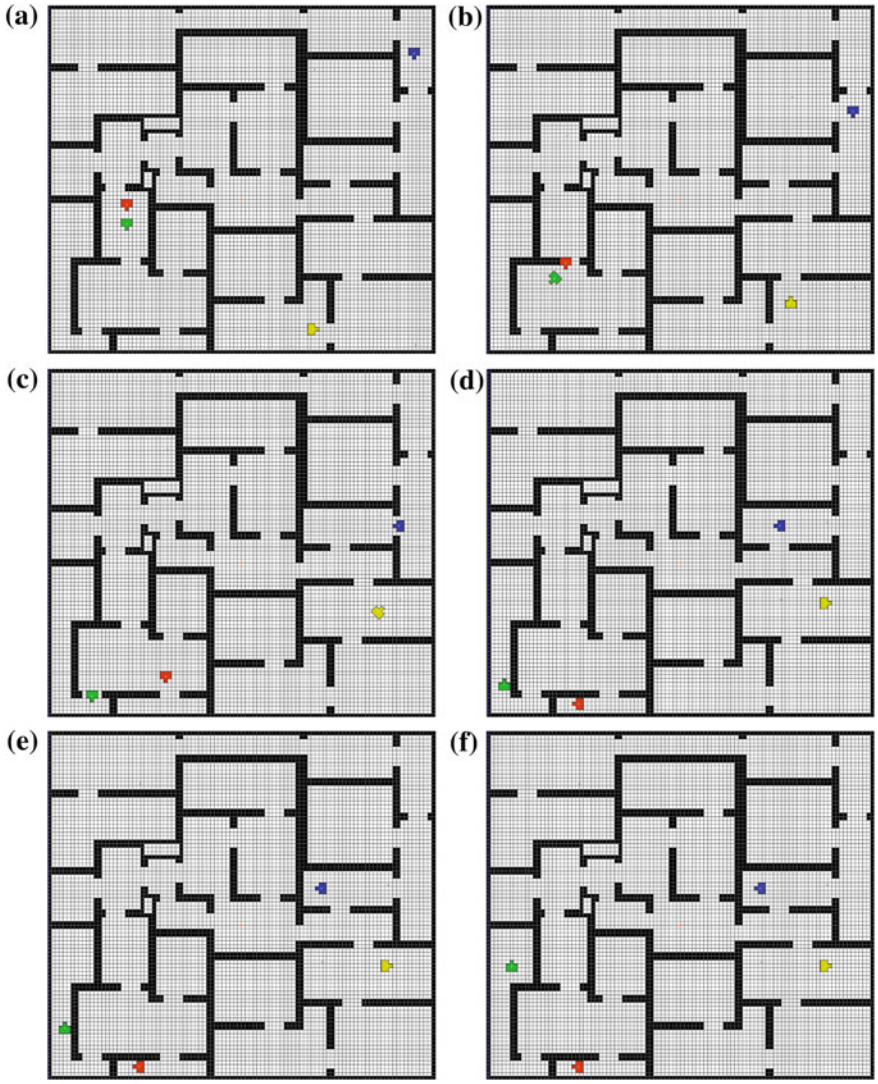


Fig. 7.21 MRS movement (2): **a** T = 30; **b** T = 45 (*Red-Green* interaction ends); **c** T = 60; **d** T = 75 (*Yellow robot* stops) **e** T = 80 (*Red* and *Blue robots* stop); **f** T = 98 (*Green robot* stops)

spatiotemporal motion planner and runs on three different microprocessors. Out of the three instances, the worst computing time results on the Yellow robot (Table 7.2), and it is the time that affects the entire procedure (0.84 s for a total of 15.1 Mcells). Therefore, the total time for all the phases is 0.85 s. On the other hand, assigning the test to only one single instance of the same coordinated motion planner handling all the four robots, the computation time required would be 4.31 s on a total of 64 Mcells

managed. Considering the planning time during the motion (real-time planning), a coordinated motion planner is 5 times slower than the solution proposed (4.31 s vs. 0.85 s).

7.6 Conclusions

A two-phase planner has been used based on a joint metric-topological representation that allows to split the general problem of planning into two separate planning issues (*divide et impera*). Clustering of cells in rectangular homogeneous areas actually introduces a multi-resolution approach: we use a more abstract topological map at a lower resolution and, in the second phase, a regular cells decomposition map at high resolution. The tessellation with variable-sized rectangles allows a complete coverage of the free space and maintains a spatial link with the highest-resolution map. This allows us to move between the two different resolution levels on a spatial basis.

The approach is characterized therefore by:

- Multi-resolution: the detection of clusters of cells leads to two resolution levels of representation, to which a third topological level is added;
- Hierarchical Planning: gross and fine planning brought on two different resolution levels, the gross one for channels identification and the fine one for the precise movements.

The gross part allows to identify channels and significantly reduce the area of interest (number of cells) operated by the spatiotemporal planner. The latter is able to determine the precise trajectories, spreading potentials only within channels. It is also able to solve the interaction problems (in the conflict areas) only between the robots involved, significantly reducing the total amount of computational time.

References

1. Barraquand, J., Langlois, B., Latombe, J.C.: Numerical potential field techniques for robot path planning. IEEE Trans. Syst. Man Cybernet. **22**(2), 224–241 (1992)
2. Behring, C., Bracho, M., Castro, M., Moreno, J.A.: An algorithm for robot path planning with cellular automata. In: Bandini, S., Worsch, T. (eds.) Theoretical and Practical Issues on Cellular Automata, Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry, Karlsruhe, D, Oct 4–6, 2000, pp. 11–19. Springer, Berlin (2000)
3. Bennewitz, M., Burgard, W., Thrun, S.: Optimizing schedules for prioritized path planning of multi-robot systems. In: ICRA, pp. 271–276. IEEE Computer Society (2001)
4. Brooks, R.A.: Solving the find-path problem by good representation of free space. In: Waltz, D.L. (ed.) Proceedings of the National Conference on Artificial Intelligence. Pittsburgh, PA, Aug 18–20, 1982, pp. 381–386. AAAI Press (1982)
5. Canny, J.F.: The Complexity of Robot Motion Planning. MIT Press, Cambridge (1988)

6. Culberson, J.C.: Sokoban is pspace-complete. In: Proceedings of the International Conference on Fun with Algorithms (FUN98), Ontario, Canada. Carleton-Scientific (1998)
7. Ioannidis, K., Sirakoulis, G.C., Andreadis, I.: A cellular automaton collision-free path planner suitable for cooperative robots. In: Panhellenic Conference on Informatics, PCI 2008, August 28–30, 2008, Samos Island, Greece, pp. 256–260. IEEE Computer Society (2008)
8. Jahanbin, M.R., Fallside, F.: Path planning using a wave simulation technique in the configuration space. In: Gero, J.S. (ed.) Artificial Intelligence in Engineering: Robotics and Processes. Computational Mechanics Publications, Southampton (1988)
9. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
10. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge (2006)
11. LaValle, S.M., Hutchinson, S.: Optimal motion planning for multiple robots having independent goals. IEEE Trans. Robot. Automat. **14**(6), 912–925 (1998)
12. Lozano-Pérez, T.: Spatial planning: A configuration space approach. IEEE Trans. Comput. **C-32**(2), 108–120 (1983)
13. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
14. Marchese, F.M.: Cellular automata in robot path planning. In: Proceedings of the First Euromicro Workshop on Advanced Mobile Robot, Kaiserslautern, D, pp. 116–125 (1996)
15. Marchese, F.M.: A path-planner for generic-shaped non-holonomic mobile robots. In: Proceedings of European Conference on Mobile Robots (ECMR 2003), Radziejowice, Poland (2003)
16. Marchese, F.M.: Multiple mobile robots path-planning with mca. In: International Conference on Autonomic and Autonomous Systems (ICAS 2006), California, pp. 56–61. IEEE Computer Society (2006)
17. Marchese, F.M.: Time-invariant motion planner in discretized c-spacetime for mrs. In: Yasuda, T. (ed.) Multi-Robot Systems. Trends and Development, pp. 307–324. InTech, Vienna, A, A (2011)
18. Parker, L.E.: Cooperative motion control for multi-target observation. In: Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97, Sept 7–11, 1997, Grenoble, France, pp. 1591–1597 (1997)
19. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Proceedings of the 20th Annual Symposium on Foundations of Computer Science, SFCS '79, pp. 421–427 (1979)
20. Tzionas, P.G., Thanailakis, A., Tsalides, P.G.: Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. IEEE Trans. Robot. Automat. **13**(2), 237–250 (1997)
21. Warren, C.: Multiple robot path coordination using artificial potential fields. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 500–505 (1990)
22. Zelinsky, A.: Using path transforms to guide the search for findpath in 2d. Int. J. Robot. Res. **13**(4), 315–325 (1994)

Chapter 8
Autonomous Robot Path Planning Techniques Using Cellular Automata

Konstantinos Charalampous, Ioannis Kostavelis, Evangelos Boukas, Angelos Amanatiadis, Lazaros Nalpantidis, Christos Emmanouilidis and Antonios Gasteratos

Abstract Path planning for autonomous navigation remains an active research topic, ensuring safe maneuver of mobile robots in congestive environments by producing collision free trajectories. To this end, this chapter introduces the amalgamation of path planning techniques with Cellular Automata (CA) operations in order to embody analogous desired properties. These navigation systems are consist of signee scene and global map components. Regarding the single scene method presented here, it achieves obstacle free routes from the current position of the robot towards the goal one. The latter serves as local planner and is accomplished via the attenuation of a v-disparity image formed upon the respective depth map. Afterwards, a CA floor field is created revealing the traversable regions within the scene. Once the target point is brought to light, a supplementary CA routine is performed on the floor field to expose the traversable route. Concerning the global map operations, a global path planning algorithm is demonstrated, suitable for dynamically changing

K. Charalampous (✉) · I. Kostavelis · E. Boukas · A. Amanatiadis · A. Gasteratos
Department of Production and Management Engineering, Democritus University of Thrace, Vas. Sofias 12, Xanthi, Greece
e-mail: kchara@pme.duth.gr

I. Kostavelis
e-mail: gkostave@pme.duth.gr

E. Boukas
e-mail: evanbouk@pme.duth.gr

A. Amanatiadis
e-mail: aamanat@ee.duth.gr

A. Gasteratos
e-mail: agaster@pme.duth.gr

L. Nalpantidis
Department of Mechanical and Manufacturing Engineering, Aalborg University, 2450 Copenhagen SV, Denmark
e-mail: lanalpa@m-tech.aau.dk

C. Emmanouilidis
ATHENA Research and Innovation Centre in Information, Communication and Knowledge Technologies, Athens, Greece
e-mail: chrisem@ceti.gr

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_8

environments. This method incorporates the A* search methodology tuned with CA strategies taking advantage of their discrete nature and, thus, administering a valuable searching approach. Moreover, the corresponding global path planning approach guarantees an obstacle free and low cost productive path. The main characteristic of this approach is that extends the map state space susceptible to time. The respective time intervals are adjusting and foresee the probable enlargement of obstructions. The performance of the examined algorithms has been evaluated on real data exhibiting remarkable results.

8.1 Introduction

Over the last years, multiple attempts were made with respect to the advancement of efficient methods for safe autonomous robot navigation. The latter comprises an effective research area in multiple scientific fields, the majority of which focuses on the formation of a collision free route from a starting point to a target one, given precise operating constraints and scenarios. The most ordinary, however significant criteria such a system should retain are the capability of obstacle avoidance, the capacity to deploy in static or dynamic environments and the computational efficiency. The derivation of a collision free path is an indispensable characteristic in any autonomous system [1]. The functionality in static or dynamic environments concern whether the world model stays intact during time advancement or not, respectively. Computational efficiency is an additional requested feature for constructing an autonomous system that finds applications in real time operations. Such constraints a system should hold have given birth to a plethora in number and variations of the proposed methods, that have guided to significant improvements especially in robot and car navigation [2], ambient assisted living [3], sensor fusion [4], logistics [5] or even in search and rescue robots [6, 7]. However, the autonomous navigation field should not be considered as a closed issue.

Concerning the estimation of traversable routes, which robots can safely follow in a scene, comprises the exposure and prevention of collision with obstacles. Consequently, it constitutes the primary step that leads to higher level navigation algorithms, namely the global path planning and simultaneously localization and mapping (SLAM). CA have been demonstrated their capabilities to offer trustworthy solutions to many robotics-related problems. Their profitable engage may be witnessed in the area of mobile robotics [8], as well as in other manufacturing and fabrication processing problems [9]. Most of those CA applications in robotic problems assume a cell grid workspace, where the robot and the obstacles acquire punctual representation [10, 11]. Namely, in the work of Marchese et al. [12], a CA method is proposed providing a solution to the path planning of a rigid body in a planar workspace where prior information for the localization of the obstacles is assumed. As a result, the mobile platform is designed to be a point that traverses from one cell to another, while the important restriction is that the robotic platform follows a smoothed trajectory without stopping and rotating with a minimum radius of curvature. Additionally, in

Behring et al. [13] suggested that the input space is a frame acquired by a camera, attaining information related to the location of all items and the coordinates of the starting and the goal point. This CA based method functions in two distinct stages taking advantage of the Manhattan distance yielding thus the most efficient path from the starting to the goal position. A different approach is followed in [14], in a nutshell, a CA technique is used to surpass the perspective-effect. This method is embodied in a vision system of an autonomous platform intended to function in indoor environments. Moreover, in [15], a CA algorithm is exploited in order to construct a real time path planning system of cooperative robots confirming that precise collision-free paths can be formed having low computational cost. A simulation algorithm that relies upon CA advances for a qualitative description of a structured indoor environment, comprising thus the stepping stone for a SLAM framework presented in [16]. Furthermore, the work in [17] proposed a planar CA component encapsulated in a SLAM algorithm. In more detail, a CA engages in the improvement of the global and local map blending action, thus, bettering the quality of the constructed map. Regarding the global path planning, it produces a variety of solutions that cope with the majority of the previously mentioned constraints. Techniques that rely upon the optimization form a set of inequalities and constraints utilized to solve the problem. Yet, with the aim to be accurately formed, the set of inequalities grows proportionally and as a result derives complex non linear functions that have to be optimized [18]. The works in [19, 20] employ the comprehensive utilization of Voronoi diagrams having the drawback of significant computational cost. The authors in [21] also exploit the Voronoi diagrams, yet their method is unable to cope with robots that attain arbitrary shapes. The work in [22] presented a technique that replicates the Dubin's theorem, employing the genetic algorithms in order to extract the optimal solution in a static environment. Another family of methodologies rely upon heuristic functions [18], with the aim to prune non-traversable paths while at the same time deal with the robot's embodiment. The work in [23] presented a technique that mimics a wave, the later generates a wave that carries the respective distances from the goal point. It was further expanded in [24], named path transform, which appends distances from adjacent obstacles into the calculations. Last, CA confirmed to be suited to administer accurate solutions to a collection of robotics-related problems. In more detail, CA have been favorably applied in the area of mobile robotics [25, 26] and in the one of robotic vision [27]. A rather significant characteristic in the larger part of CA applications in robotics is the assumption of a cell grid workspace, in which the robot and the obstacles are punctual [10].

The chapter in hand introduces two methodologies, dealing individually with the local and path planning problem, respectively. Both methods provide solutions for the respective navigation components which are thoroughly discussed and experimentally evaluated. In short, the local path planning solution discussed in a subsequent section applies 3D vision methodologies and CA operations in order to derive a collision-free path for a robot in a scene. In the first step, a stereo vision method or a suitable depth sensor is utilized for the formation of the disparity map of the respective scene. The second step includes the employment of a polar transformation on the disparity map to create a polar-depth frame, in which the objects are

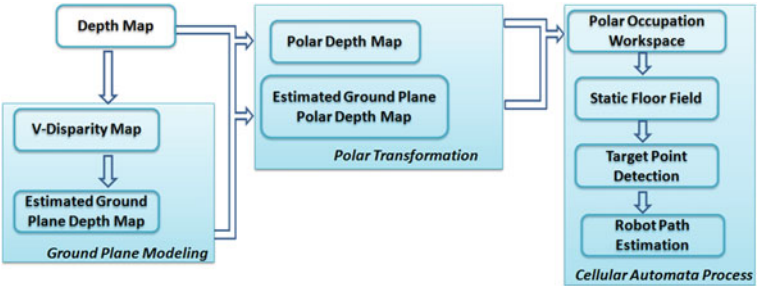


Fig. 8.1 The flow chart of the proposed methodology

readjusted with respect to an angular formation. Then, a v-disparity routine models the ground plane of the scene considering the original depth frame and, as a result, a novel obstacle-free depth map is computed. The latter map is also converted into a polar one. Afterwards, the comparison of the polar maps takes place and extract a binary polar workspace uncovering the presence of objects towards all possible directions. The successive step involves the creation of a floor field that relies on CA operations and comprises the computation of the distances among the positioning of the robot and the traversable areas in the scene. Last, the target position is discovered within the scene and via the exploitation of CA rules over the floor field, the obstacle free path is calculated. Figure 8.1 illustrates the fundamental acts of the described technique.

The global path planning algorithm analyzed in this chapter pledges a traversable route among obstacles that are capable of presenting a dynamically expanding behavior. At the same time, the respective path cost produced is locally optimal one in all the intervals. The extracted overall trajectory is found to be in the vicinity of the global optimum due to the fact that attempts to acquire the global optimum, in every iteration. The inflation of the dynamic obstacles rely on particular CA rules [28]. For that reason, in conjunction with the corresponding dynamic obstacles various expansion rules may be performed. The method described here considers the Moore's expansion neighborhood which is the one that wraps more area and leads to a more generalized solution. A demonstration of the discussed method also takes place within this chapter. The obtained 3D map is then top-down projected to create the 2D workspace upon which the CA operations take place. The presented methodology is defined as a computational efficient one, suitable for real time operations.

8.2 Theoretical Background

8.2.1 Cellular Automaton Theory

Automaton is a mathematical structure, outlined in a formal fashion as a quintuple:

$$\{I, Z, Q, \delta_1, \omega\} \tag{8.1}$$

where I, Z are the two sets of the corresponding inputs to the defined automaton, Q is a set of internal states, and δ_1 is a transition function defined as δ_1: $I \times Q \rightarrow Q$, that derives the respective internal state from the Cartesian product of an internal state and the set of inputs. Moreover, ω is a function that extracts the output from the Cartesian product of an internal state and the set of inputs, defined as ω: $I \times Q \rightarrow Z$.

Cellular automata is a subdivision of automata, having as a main property the capability to handle simultaneously with time and space in a discrete fashion. According to the latter definition, a subsequent state is the derivative of the exact previous one. However, this statement can be extended, guiding to a system with memory, capable of inferring decisions by examining several previous states. The spacial region around a cell is characterized as neighborhood, while a cell is in position to explore it. Regarding the size of a neighborhood is theoretically unbounded, though it needs to be the same for all cells. Two well known classes of neighborhoods are the Von Neumann and the Moore ones. The aforementioned one attains a diamond shape illustration in the case of a square grid. Let a cell (x_0, y_0) in a two-dimensional plane, and length r, Von Neumann neighborhood is defined as [29]:

$$N^V_{x_0,y_0} = \{(x, y) : |x - x_0| + |y - y_0| \leq r\} \tag{8.2}$$

However, Moore's neighborhood derives a squared shaped neighborhood in a squared grid. Let a cell (x_0, y_0) in a two-dimensional plane and length r, Moore neighborhood is defined as [29]:

$$N^M_{x_0,y_0} = \{(x, y) : |x - x_0| \leq r, |y - y_0| \leq r\} \tag{8.3}$$

8.2.2 Path Planning Theory

The field of path planning is separated in global and local one. Global path planning (GPP) and local path planning (LPP) are two complementary functionalities for autonomous navigation. On the one hand, GPP operates in a high level, performs long term planning, seeks for the shortest path to a goal according to certain criteria and avoids "cul-de-sacs" situations. On the other hand, a local path planner focuses on smaller obstacles lying close to the robot's sensory input and derives a feasible, continuous, obstacle free trajectory in a time efficient fashion. Both problems are thoroughly studied individually, yet the recent notion is the proposition of hybrid methodologies, i.e. systems that provide simultaneous global and local resolutions.

A well-known component that has been widely used for global path planning is the A* search algorithm. The latter is a search algorithm, well known one in computer science, primarily in path seeking and graph traversing problems. A* was presented as a continuation of Dijkstra's algorithm [30]. Initially, it was suggested for two-dimensional grids, referred to as state spaces, indicating its discrete nature. A* may

be called as the mixture of Dijkstra's algorithm since it can retrieve the shortest path and of the Greedy Best-First-Search due to the fact that utilizes a heuristic function. In more detail, A* pursues the path that minimizes the cost function $f(s)$, where:

$$f(s) = g(s) + h(s) \tag{8.4}$$

The $g(s)$ part demonstrates the cost of the already traversed path until the present state-cell s. The $h(s)$ path represents a heuristic function that supplies with an appraisal of the cost from the present node to the goal one. In the case where $h(s)$ is an admissible function, i.e. $h(s)$ never overestimates the true cost from current state to goal providing a value of less or equal to real cost, then A* attains several intriguing features, namely the fact that the extracted path will definitely be the shortest-optimal one [31]. From the very beginning of A*, a variety of heuristic functions have been presented over the last decades, providing adequate solutions with respect to the nature of a certain problem. A typical characteristic among all those methods is the trade-off between heuristic's accuracy and time complexity in order to produce a result, in the form of an estimation. The key point in order to convert a search method suitable for real time applications may be found in an adept heuristic function. With respect to the nature of each problem, the adopted plan of action should be regarded as the optimal trade off between global optimum path cost and fast convergence. Moreover, A* is a complete method, i.e. it constantly spans a solution when such one is in existence.

The computational complexity of A*, as noticed before, depends upon the utilized heuristic function $h(s)$. The complexity is bounded to be polynomial in the case where the following restrictions hold: (i) State space is a tree, and (ii) for a given heuristic function $h(s)$ the next condition is valid:

$$|h^*(s) - h(s)| = O(\log(h^*(s))) \tag{8.5}$$

where, $h(s)$ is the heuristic's value at state s, and $h^*(s)$ is the optimal function's value at state s, i.e. the actual cost from state s to goal. The latter suggests that the approximation error, expressed as the absolute difference between the optimal function and the heuristic of interest, needs to be equivalent to the optimal solution's logarithmic value. Yet, the number of nodes may similarly be extended in an exponential fashion, while extracting the minimum cost route [32].

8.3 Local Path Planning

8.3.1 Depth Map Acquisition

The initial step of the presented local path planner attains the calculation of a disparity frame, which embodies the depth information of the scene [33]. Although the construction of a stereo vision method that derives disparity frames is out of the scope of this chapter, it is worth mentioning that there is a variety of techniques regarding

the computation of the depth information comprising miscellaneous depth sensors and techniques. Several cameras exist, capable of attaining a continuous stream of depth frames namely *Microsoft Kinect* or *Time of Flight Cameras* (ToFC) which are nowadays generally available, while several research has already been done in this area [34, 35]. Moreover, a famous method capable of attaining the depth of a scene is the usage of a stereo correspondence system, which acquires as input a pair of frames captured from a stereo rig and derives the corresponding disparity map [36]. The methodology presented here does not assume a specific type of depth map acquisitions, as long as the respective of derives accurate and dense depth maps. In the absence of a depth estimation camera system, an advisable algorithm for robotic applications that deals with the stereo correspondence is presented in [37]. The capabilities of this stereo algorithm has been thoroughly tested in real world navigation systems [17], the output of which is illustrated in Fig. 11b.

8.3.2 Obstacle Free Ground Plane Modelling

An acquired depth map, similar to the one obtained from the aforementioned stereo correspondence algorithm, may be utilized for the calculation of a v-disparity image, as depicted in Fig. 8.2c. The latter considers a horizontal histogram of the disparity values, involving fundamental geometrical attributes of the respective scene and it administers, in a straightforward manner, an assessment of the horizon and the obstacles over the ground plane. In a v-disparity image, each pixel has a positive integer value that denotes the number of pixels in the input image that lie on the same image line (ordinate) and possesses disparity values equal to its abscissa [38]. The knowledge acquired from the v-disparity image is essential since it separates the pixels corresponding to the ground from the ones referring to objects. The ground plane may be expressed in terms of a linear equation $f(x) = \alpha \cdot x + \beta$ in a disparity image, in which the variables α and β refer to the slope of the camera and to the height that the ground plane meets the horizon in the scene respectively. The aforementioned

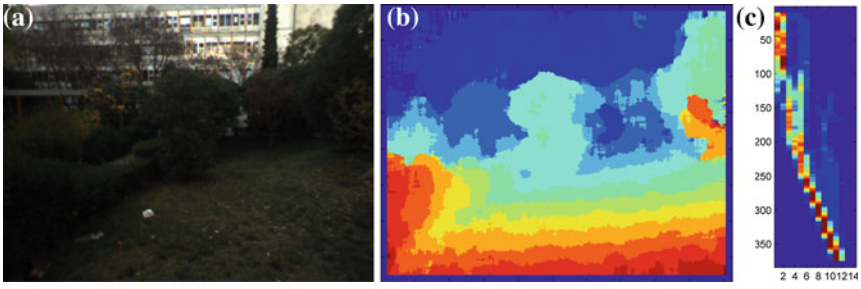


Fig. 8.2 **a** *Left* reference image of the stereo pair, **b** The disparity map, **c** The calculated v-disparity map

parameters may be calculated via the Hough transform [39], where the geometrical attributes of the camera-environment framework are assumed to be known. Afterwards, the utilization of α and β aids to the reconstruction of a null-disparity image, resembling to an object free frame, having a fundamental geometry akin to the one of the original depth map.

8.3.3 Polar Transformation of the Depth Map

The original and the assessed ground plane depth map are remapped via the usage of a polar transformation, where the latter readjusts the knowledge of the depth in the scene and positions the disparity values in a radial topology in the circumference of the robot. As a result, a spatial arrangement of the objects, positioned around the robot is attained. This transformation results two separate polar depth frames. The first one is one is the *polar-depth* information extracted from the transformation of the original disparity frame (Fig. 8.3b) and the second one is the *ground plane polar-depth map* (Fig. 8.3d) derived from the transformation of the estimated obstacle-free depth map. Every column of the polar-depth map refers to a certain direction in the scene, for example the 90th column refers to a direction at 90°, that is straight ahead of the robotic platform. The most essential asset of the applied polar-depth mapping is the fact that the depth information is favorably readjusted in order to refer to the probable directions the robot can move towards to. As a result, every column characterizes the spatial distribution of the obstacles being on the corresponding direction. In more detail, the pole, from which the polar coordinate system initiates, is placed to the central pixel of the bottom row of the disparity image $O(M, \frac{N}{2})$, where M and N designate the number of the disparity rows and columns respectively (Fig. 8.3b). The chosen radial resolution ρ guarantees that the knowledge of the image is uniformly distributed in the polar space, while the angular resolution θ is a variable obtaining values within the range $[0°, 180°]$ with one degree step. Afterwards, the two polar-depth maps are subtracted (Fig. 8.3e) in order to derive a binary frame where the areas referring to the ground plane are marked with zeros (0), while obstacles are marked with ones (1) (Fig. 8.3f). To be more accurate, the outcome of the subtraction is further thresholded because the *ground plane polar-depth map* comprises an estimation based on the v-disparity frame.

8.3.4 Floor Field

The binary polar frame comprises the area where the floor field for every pixel (i, j) is formed [40]. The latter field comprises a gradient that attains low values adjacent to the robot and high ones as the distance from it grows [41]. The dimensionality of the respective area is equivalent to the original polar-depth maps and, as a result, the $F_{i,j}$ space is populated according to the subsequent rule:

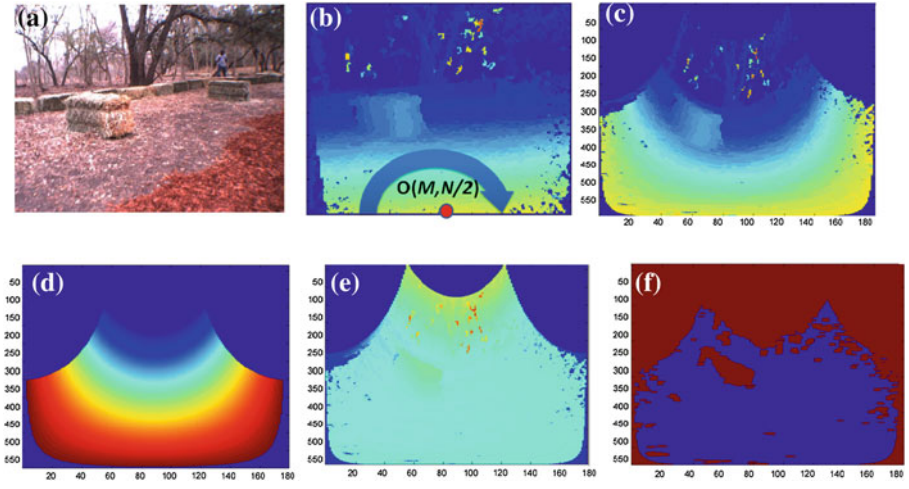


Fig. 8.3 **a** The *left* reference image, **b** the depth map, **c** the polar-depth map, **d** the reconstructed *ground plane polar-depth map*, **e** the difference of the two depth maps, **f** the binary image indicating the obstacles

$$F_{i,j} = \begin{cases} >0, & \text{if a distance value has been assigned} \\ =0, & \text{if no value has been assigned yet} \\ =-1, & \text{if there is an obstacle.} \end{cases} \quad (8.6)$$

The calculation of the floor field relies on a CA operation that considers the Von Neumann neighborhood. As already discussed, the latter is a diamond shaped one, utilized in order to describe the set of cells around the principal one in the middle, having coordinates (i_0, j_0). In the described methodology, the Von Neumann neighborhood is alter in order to leave aside the central cell. Additionally, the CA operates in a parallel fashion with respect to a particular rule, which is repeated until all the $F_{i,j}$ cells attain non zero values. At every time instance t the CA rule is performed upon all cell with null values $F_t(i_0, j_0)$ and considers the values of the neighbor pixels $F_t(i, j)$ within the Von Neumann one. Correspondingly, in the subsequent time step $t+1$ the corresponding cell $F_{t+1}(i_0, j_0)$ acquires the minimum positive value of the respective neighborhood, grown by one explained in Eq. 8.7, while in a contrasting case the value of the cell a time $t+1$ remains equal to the one in t (Eq. 8.8).

$$\textbf{IF } F_t(i_0, j_0) = 0 \textbf{ AND } F_t(i, j) > 0 \textbf{ THEN } F_{t+1}(i_0, j_0) = \min F_t(i, j) + 1 \quad (8.7)$$

$$\textbf{IF } F_t(i_0, j_0) = 0 \textbf{ AND } F_t(i, j) \leq 0 \textbf{ THEN } F_{t+1}(i_0, j_0) = F_t(i_0, j_0) \quad (8.8)$$

where (i, j) belongs to the Von Neumann neighbor. Figure 8.4a illustrates a traversable scene, which encompasses several objects and it is transformed into a polar binary one (Fig. 8.4b). The latter rule is performed upon the binary frame and the composed floor field is depicted in Fig. 8.4c. In this figure, the areas that refer to in obstacles have been omitted.

8.3.5 Local Path Estimation

After the calculation of the floor field, the goal position in the respective scene need to be chosen. The latter position is the most distant point that the robot can traverse to. The binary polar workspace is exploited in order to compute the distance in terms of pixels between the position of the robot and the goal one. In more detail, the number of the row that the goal point is found in the polar-depth map, suggests the longest distance the robot is capable of traversing. Based on the fact that this section deals solely with traversable areas in the scene the acquisition of the goal position is limited within the area that resembles the horizon. The latter is already calculated (β in the v-disparity), and, as a result, the maximum number of the rows in the polar depth map that analyzed is equal to β as illustrated in Fig. 8.5a. The subsequent step involves the retrieval of the path the robot has to pursue in order to be found to the horizon of the scene. Having that aim, a supplementary CA operation has been considered taking advantage of the pre-calculated floor field, which embodies the

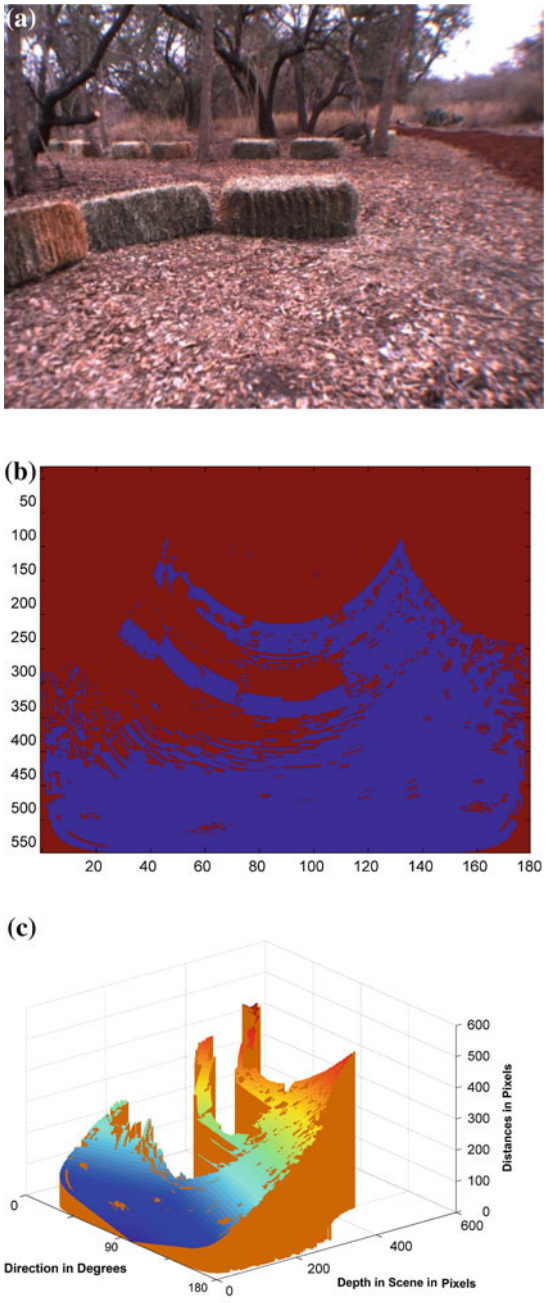


Fig. 8.4 **a** A reference image of a scene, **b** the polar workspace that contains the obstacles and the ground plane, and **c** the respective floor field

sum of distances from the position of the robot to all traversable points. As a result, seeing as a initial point the goal position and via the transiting to the cell that obtain lower distance values a trace which connects this point with the robot location can be calculated. This trace reveals the chosen route and the corresponding calculation relies to the CA via the application of the Moore neighborhood, which is a square shaped one. The workspace of the CA operation consists of the already calculated floor field and the binary polar image. Yet, it is investigated solely up to the curve that refers to the horizon, as predicted by the v-disparity. Fore that reason, the novel planar workspace $P_{i,j}$ is described with respect to Eq. 8.9.

$$P_{i,j} = \begin{cases} =0, \text{ traversable region} \\ =1, \text{ occupied region} \\ =2, \text{ path indexes.} \end{cases} \tag{8.9}$$

Firstly, the goal position on the workspace is marked solely, as *path index*, and then, the route that the robot need to pursue in order to maneuver among the obstacle is extracted via the application of the subsequent CA rule:

$$\textbf{IF} \sum_{i=-1}^{1} \sum_{i=-1}^{1} P_t(i, j) < 3 \ \textbf{THEN} \ P_{t+1}(i_{min}, j_{min}) = 1 \tag{8.10}$$

where the (i_{min}, j_{min}) are the indexes that correspond to the minimum values in the respective neighborhood in the floor field, i.e. $[i_{min}, j_{min}] = \text{argmin} F(i, j)$. Additionally, the CA operation is accomplished at the same time in accordance with the previously described rule, until the path that joins the location of the robot and the horizon is concluded. The calculated path which exploits the CA rule over the respective floor field is illustrated in Fig. 8.5b. Moreover, the pixels that refer to the polar path are transformed back to the Cartesian coordinate system and, as a result, the calculated path is depicted in the reference frame in Fig. 8.5c.

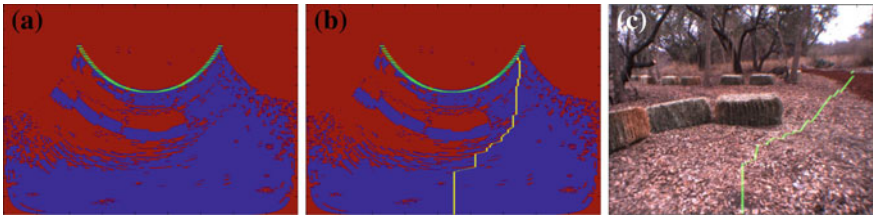


Fig. 8.5 **a** A binary polar image with the horizon, **b** the estimated path in the polar space, **c** the respective path in the Cartesian space

8.4 Global Path Planning

8.4.1 Operation in the Continuous Space

The method discussed in this section relies on the A* algorithm due to the fact that takes advantage of the CA theory. The aim of this algorithm is the extraction of a kinematically feasible path to a target position with rapid convergence and at the same time considering the formation of dynamic obstacles and their expansion. Due to the fact that the A* search algorithm may be performed on two-dimensional grids, similar to the fundamental CA properties of time and space discrete level of processing, permits a seamless blending of both CA and A* theories. As mentioned previously, the he A* algorithm as originally presented, is endowed with several interesting properties, however they can only be performed upon a static and completely observable grid maps. With the aim to alleviate the previously mentioned disadvantage, the discussed algorithm constructs a seamless union between A* and CA theories. The fundamental idea relies on the fact that since A* in not applicable on dynamic grids, it will be preformed on anticipated grid maps where the formation properties and the growth of the obstacles are taken into consideration. The prognosis of the grid map is possible due to the discrete nature of both A* and CA theories. Yet, is necessary to apply a time step. This limitation is solved by utilizing an adaptive step. This method sets the time step in an iterative fashion as the grid map evolutes. The time step is calculated as follows:

$$D_o(i, j) = \frac{dist_o(O_i, O_j)}{2} * r_e(i) \qquad i, j = 1, \ldots, N_o, \qquad i \neq j \tag{8.11}$$

$$V_r(i) = \frac{dist_r(r, O_i)}{2} * r_e(i) \qquad i = 1, \ldots, N_o \tag{8.12}$$

$$T_s = min(D_o, V_r) \tag{8.13}$$

where N_o is the number of obstacles, O_i defines the coordinates of the i-th obstacle, $r_e(i)$ is the expansion rate of obstacle i, $dist_o$ is an operator that calculates the Euclidean distance between two obstacles and $D_o(i, j)$ is a matrix. Let the number of obstacles be N_o, the methodology computes the mid distance between two obstacles and then multiplies it with the growth rate of every one of them. In every time step, the matrix $D_o(i, j)$ attains the time interval that is essential in order for an obstacle O_i to reach half the distance to the obstacle O_j. In (8.12), $dist_r$ is an operator that calculates the naive Euclidean distance between an obstacle and the robot. Vector $V_r(i) \in \mathbb{R}^{N_0}$ stores the time in which an obstacle O_i will reach half the distance to the robot. In (8.13), min is an operator that finds the minimum time for any of the above events to happen defining the time step T_s. The definition of a time step is accompanied with the evolution of the map T_s steps in time. As a result, A* is performed upon the novel map and progresses the robot T_s time steps. Then, follows the recalculation on a new time step. Those step are iterated until the robot is found to the target position. The justification of the time step computation may

be found in the fact that the obstacles in the anticipated map will avoid collision, resulting constantly to a kinematically feasible between them, for the robot to follow. Regarding the case where the time interval between an obstacle and the robot is less that the necessary time for two obstacles to collide, this time interval is selected to be the new time step, guaranteeing the avoidance of impact with respect to the robot. In the discussed algorithm, the expansion of the obstacles obey the Moore's neighborhood rule, without being a constraint, since all types of neighborhoods may be preformed to the presented strategy. Yet, Moore's neighborhood can be regarded to be as the most disastrous way an obstacle can expand, in terms of coverage upon the grid map. The expansion rate may vary among the obstacles, while in this algorithm it is also allowed the addition of obstacles during execution, as well as the alteration of the target location. The latter may be applied previous to any iteration, where the recalculation of the path takes place, absent of additional computational burden. In order to guarantee the computation of a safe path, the A* algorithm is performed, at a current time t by exploiting the anticipated grid map of time interval $t + T_s$. In this fashion, the A* behaves towards the obstacles on the anticipated map as being the ones in the present one. The projection of the expansion of obstacles with respect to time provides the assurance that the computed trajectory will be always kinematically. With the aim to decrease the computational burden, the presented algorithm screens whether the subsequent conditions are valid:

1. The imaginary line segment that joins two obstacles does not intersect with the robot path,
2. If an obstacle has a sufficiently greater expansion rate than another, leading at a time t_o the second obstacle to be overlaid from the first one, then only the first obstacle needs to be taken into account from time t_o onwards.

The computational complexity of this algorithm is in proportion with the number and location of the obstacles. The subsequent cases are presented to exhibit the A* method's fluctuations. Considering the most unfavorable case, where all obstacles need to be considered, the computational complexity is $O((N_o)^2)$, where N_o defines the number of obstacles. In the best case, where the robot distance from the goal can be traversed in less time than the time step T_s, then the complexity is reduced to that of A*. The most frequent case suggest that k obstacles comply with the previously mentioned conditions, and, thus, only $N_o - k$ obstacles are considered in the loop, leading the computational complexity to be equivalent to $O((N_o - k)^2)$.

8.4.2 From the Continuous to the Discrete Space

The introduced method has been evaluated on real-world data which are attained using a mobile robot platform. The respective setup consists of the wheeled robot ERA-Mobi with and on-board pc bearing an Intel Core2Duo at 2.0 GHz processor and 4GB of RAM. The servo-electric rotary pan-tilt actuator PW-70 by SCHUNK has been adopted, which cooperating with the SICK LMS500 PRO 2D laser scanner

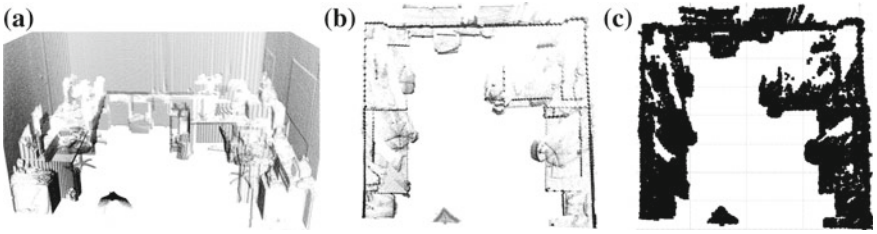


Fig. 8.6 **a** A 3D point cloud resulted from an indoors environment, **b** the respective *top-down* projection, **c** the derived grid map on which the proposed path planning methodology is performed

produces the resulted 3D point cloud as depicted in Fig. 8.6a. The reconstructed space has an area equal to $4.7 \times 4.4\,\mathrm{m}^2$ and refers to the continuous scene. Once the ceiling of the area is removed from the point cloud, then it is top-down projected (Fig. 8.6b). The subsequent step consist of the adaptation of continuous 2D map to a discrete 2D grid, upon which the presented technique may be applied. The acquired occupancy grid is illustrated in Fig. 8.6c, where every cell refers to 4 mm administering sufficient resolution for the path planner to extract dependable and feasible route.

8.5 Experimental Evaluation

8.5.1 Local Path Planning

The performance of the presented local path planning methodology has been appraised on the "Learning Applied to Ground Robots" (LAGR) from DARPA. The latter comprises a series of natural outdoor scenes [42]. Regarding the dataset, the reference color images are accompanied with the respective disparity information. The geometry of the camera-environment was unknown and constantly variating. As a result, the ground plane in the computed v-disparity frames was modeled by considering the Hough transformation. The dataset comprises of immediate and indirectly traversable scenes and, as a result, the introduced method was tested on both scenarios. To be more accurate, among the entire tested dataset four different cases are illustrated in Fig. 8.7, in which the in-between results and the conclusion of the method are presented. The first two cases consist of the evaluation of the presented algorithm on fully traversable scenes, attaining no obstacles (Fig. 8.7c) or obstacles occur in the background (Fig. 8.7f). The third case comprises traversable scenes yet, the presence of obstacles nearby the robot occurs. The calculated path among the position of the robot and the dictated goal one, which is found in the horizon, avoids all the obstacles (Fig. 8.7i). Additionally, the last case refers to another immediate traversable scene, where the presented algorithm administers to design a path that bypasses all the obstacles within the scene (Fig. 8.7l).

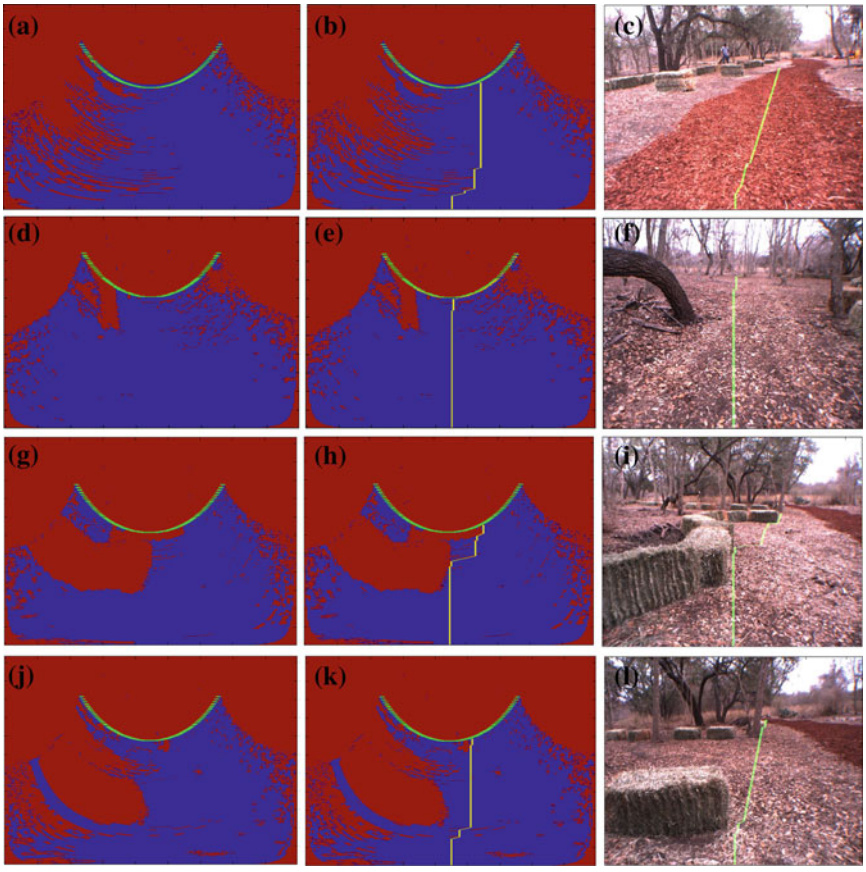


Fig. 8.7 Experimental results for four different scenarios. **a–f** correspond to a fully traversable scene, whilst **g–l** correspond to a traversable scene with some obstacles around the robot

8.5.2 Global Path Planning

The experimental evaluation of the presented global path planner takes place herealong with the robot set up- n both indoors and outdoors scenarios. Considering the evaluation, a mobile platform geared with a laser scanner has been used as depicted in Fig. 8.8. Considering the indoors scenario, the walls were constraining the area of practice. The room has an area of $4.7 \times 8\,\mathrm{m}^2$, and the original placement of the robot is a random one. At the beginning, the 3D scanning component is performed, accomplishing a 360° scan sweep. Then, the points are congregated and the 3D representation of the indoor place is concluded. Subsequently, the points that refer to the ceiling and the floor are excluded and the rest of them are top-down projected. Figure 8.9a depicts the respective 3D reconstruction, Fig. 8.9b illustrates the top-down projection and Fig. 8.9c the derived occupancy map. At this juncture the grid comprises solely static obstacles, the dynamic ones are provided individually along

Fig. 8.8 The robotic platform used in the respective work, equipped with a 3D laser scanner and a pan-tilt unit

with the goal position. Regarding the first indoor evaluation only one expanding obstacle participated, while its expansion rate was equivalent to 0.3. It is obvious that the initial trajectory of the robot is curvy, avoiding the collision. Once the traversal among obstacle is concluded the robot pursues a line, due to the A* characteristic. Figure 8.9d illustrates the initial and the goal position as well as the starting point of the expanding obstacle. An intermediate state of the path planning procedure is depicted in Fig. 8.9e. The second indoor evaluation comprises two expanding obstacles attaining different expansion rates equivalent to 0.3 and 0.4, accordingly. The initial position of the robot, the goal one and the obstacles are illustrated in Fig. 8.9g. It is apparent that the robot has to travel among the expanding obstacles, yet, as illustrated in Fig. 8.9h, when the robot surpasses the static obstacle in the middle of the room it does not continue in a line, rather seeks to maximize the margins among the expanding ones as an outcome of the proposed path planning methodology. Regarding the outdoors case, the 3D point cloud acquisition is identical to the indoors one. The most significant alteration is the fact that the scanning area is dramatically wider, concluding to an also broader grid map. Figure 8.10a, b, c depict the 3D construction, the top-down projection and the occupancy grid, accordingly. The first case (Fig. 8.10d–f) comprises two expanding obstacles with the same expansion rate of 0.4. The second case (Fig. 8.10g–i) consists of two expanding obstacles with expansion rates of 0.4 and 0.6, accordingly.

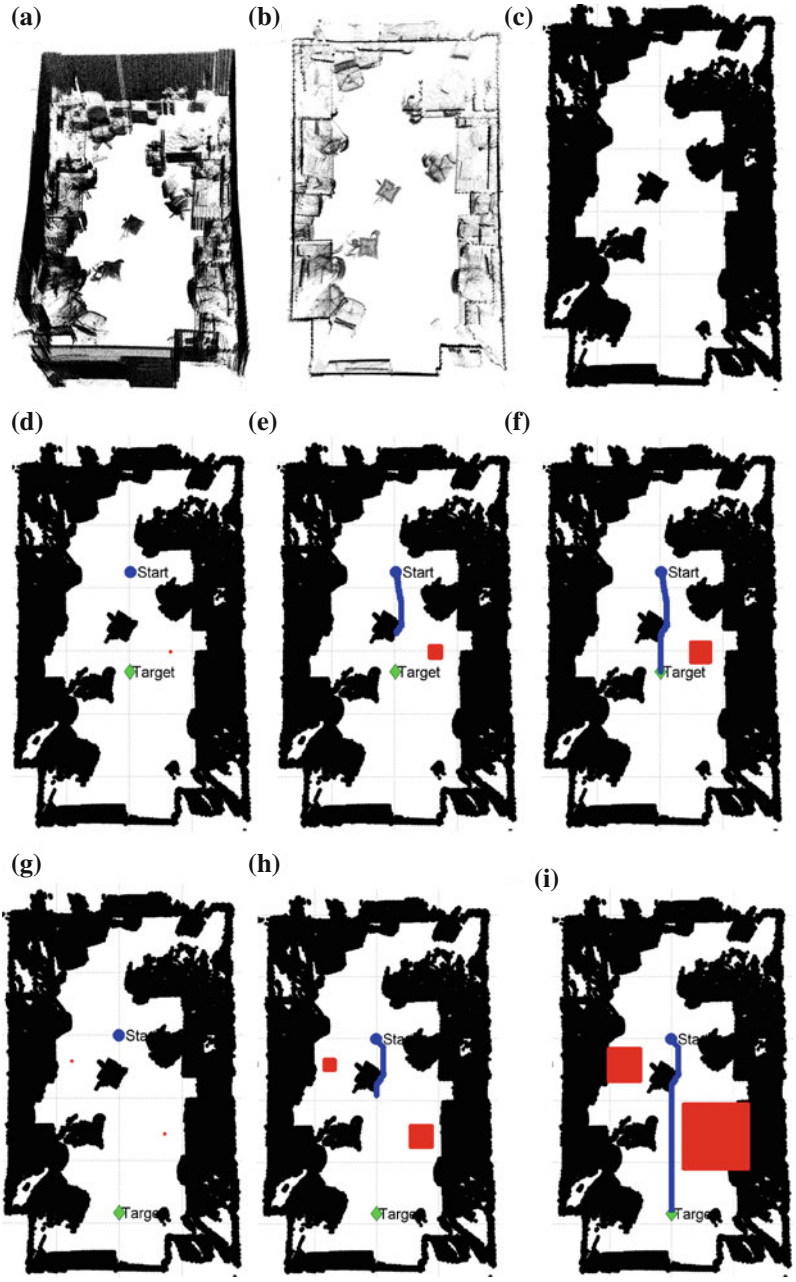


Fig. 8.9 **a** A 360° 3D point cloud resulted from an indoors environment, **b** the respective top-down projection, **c** the derived grid map on which the proposed path planning methodology is performed, **d**–**f** the derived trajectory for the corresponding start and target position, **g**–**i** the derived trajectory for different start and target position

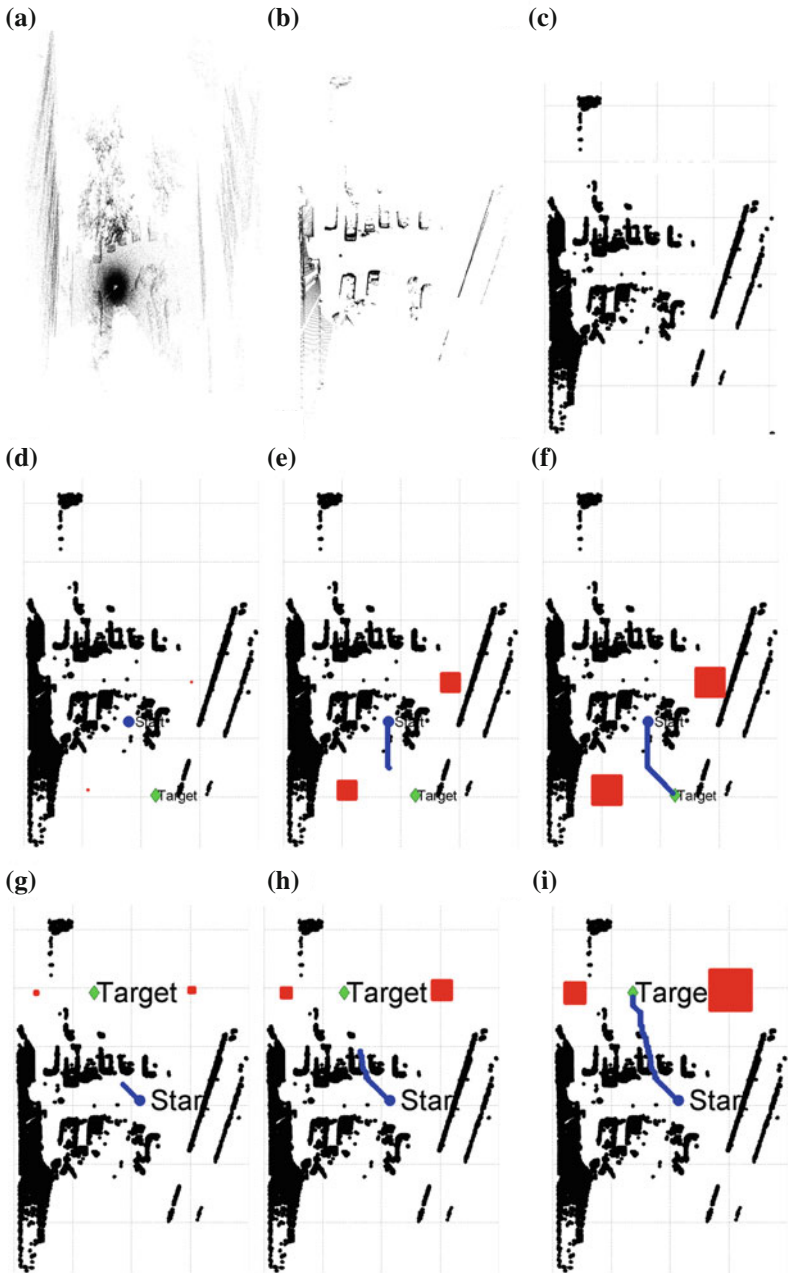


Fig. 8.10 **a** A 360° 3D point cloud resulted from an outdoors environment, **b** the respective *top-down* projection, **c** the derived grid map on which the proposed path planning methodology is performed, **d**–**f** the derived trajectory for the corresponding start and target position, **g**–**i** the derived trajectory for different start and target position

8.6 Discussion

The objective of this chapter was to present the application of the CA theory applied on local and global path planning techniques for autonomous robot navigation. The local path planning algorithm concerns the local obstacle avoidance for a robot which is equipped with a stereoscopic camera. The presented algorithm operates on the image plane and produces obstacle free routes. More precisely, it concerns a stereo vision module responsible for the depth acquisition of the scene. An auxiliary modeled depth map has been calculated utilizing the v-disparity image algorithmic procedure. Then a polar transformation is applied to the depth images producing a radial rearrangement of the obstacles in the scene. After a thresholding procedure, the polar planar workspace that contains only the obstacles of the scene is further processed by CA techniques. In the latter, the floor field is computed and the estimated path is produced after an additional CA step. The main advantage of the proposed method is that the occupancy grid is formed directly on the image plane by exploiting the polar transformation, contrary to the already existed techniques that assume a planar occupation grid as a workspace for the CA. This attribute endows the proposed method with the ability to directly track the obstacle free path within the scene. The second part of this chapter concerned a global path planner which assumes a formed 2D map of the explored environment. This algorithm is responsible to dictate a sequence of points that the robot should track in order to maneuver from its current location to a target one. The adopted strategy outlines a collision-free and a low-cost path planning framework able to provide efficient solution to robot navigation under the appearance of dynamically changing obstacles. The discrete A* algorithm along with the CA rules resulted into a strong finite tool upon which the search tactic relies. The final trajectory produced attains the neighborhood of the global optimum since it tries to find the global optimum in every iteration. The two examined algorithms have been evaluate on real world data acquired with actual mobile robot platforms. They exhibited remarkable performance in terms of accuracy, proving that the CA can provide efficient solutions in the autonomous robot navigation. Moreover, due to their nature of implementation the CA solutions are characterized by their low computational cost and their ability of parallelization, making them suitable for real-time applications.

Acknowledgments This work has been partially funded by the GSRT project ‘Polytropon’ (GSRT KRIPIS2012).

References

1. Kostavelis, I., Nalpantidis, L., Gasteratos, A.: Collision risk assessment for autonomous robots by offline traversability learning. Robot. Auton. Syst. **60**(11), 1367–1376 (2012)
2. Kostavelis, I., Gasteratos, A., Boukas, E., Nalpantidis, L.: Learning the terrain and planning a collision-free trajectory for indoor post-disaster environments. In: International Symposium

on Safety, Security, and Rescue Robotics, SSRR, pp. 1–6 (2012)
3. Balaguer, C., Gimenez, A., Huete, A.J., Sabatini, A.M., Topping, M., Bolmsjo, G.: The mats robot: service climbing robot for personal assistance. IEEE Robot. Autom. Mag. **13**(1), 51–58 (2006)
4. Kyriakoulis, N., Gasteratos, A., Amanatiadis, A.: Comparison of data fusion techniques for robot navigation. In: Advances in Artificial Intelligence, pp. 547–550. Springer (2006)
5. Jin, B., Zhang, L.: A path planning method of contingency logistics based on max-min ant system. In: 2010 Asia-Pacific Conference on Wearable Computing Systems (APWCS), pp. 311–314 (2010)
6. Amanatiadis, A., Chatzichristofis, S., Charalampous, K., Doitsidis, L., Kosmatopoulos, E., Tsalides, P., Gasteratos, A., Roumeliotis, S.: A multi-objective exploration strategy for mobile robots under operational constraints. IEEE Access **1**, 691–702 (2013)
7. Beltrán-González, C., Gasteratos, A., Amanatiadis, A., Chrysostomou, D., Guzman, R., Tóth, A., Szollosi, L., Juhász, A., Galambos, P.: Methods and techniques for intelligent navigation and manipulation for bomb disposal and rescue operations. In: IEEE International Workshop on Safety, Security and Rescue Robotics, pp. 1–6 (2007)
8. Marchese, F.: Multiple mobile robots path-planning with MCA. In: ICAS'06, 2006 International Conference on Autonomic and Autonomous Systems, p. 56. IEEE (2006)
9. Sirakoulis, G., Karafyllidis, I., Thanailakis, A.: A cellular automaton for the propagation of circular fronts and its applications. Eng. Appl. Artif. Intell. **18**(6), 731–744 (2005)
10. Zhou, K., Ma, S., Zhu, X., Tang, L., Feng, X.: Improved ant colony algorithm based on cellular automata for obstacle avoidance in robot soccer. In: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 5, pp. 298–302. IEEE (2010)
11. Marchese, F.: A directional diffusion algorithm on cellular automata for robot path-planning. Future Gener. Comput. Syst. **18**(7), 983–994 (2002)
12. Marchese, F.: Cellular automata in robot path planning. In: Proceedings of the First Euromicro Workshop on Advanced Mobile Robot, 1996, pp. 116–125. IEEE (1996)
13. Behring, C., Bracho, M., Castro, M., Moreno, J.: An algorithm for robot path planning with cellular automata. In: Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry: Theoretical and Practical Issues on Cellular Automata, pp. 11–19. Citeseer (2000)
14. Adorni, G., Cagnoni, S., Mordonini, M.: Cellular automata based inverse perspective transform as a tool for indoor robot navigation. In: AI* IA 99: Advances in Artificial Intelligence, pp. 345–355 (2000)
15. Ioannidis, K., Sirakoulis, G.C., Andreadis, I.: Cellular ants: a method to create collision free trajectories for a cooperative robot team. Robot. Auton. Syst. **59**(2), 113–237 (2011)
16. Pradel, G., Hoppenot, P.: Symbolic trajectory description in mobile robotics. J. Intell. Robot. Syst. **45**(2), 157–180 (2006)
17. Nalpantidis, L., Sirakoulis, G., Gasteratos, A.: Non-probabilistic cellular automata-enhanced stereo vision simultaneous localization and mapping. Meas. Sci. Technol. **22**, 114027 (2011)
18. Hwang, Y.K., Ahuja, N.: Gross motion planning a survey. ACM Comput. Surv. (CSUR) **24**(3), 219–291 (1992)
19. Rao, N.S.V.: Algorithmic framework for learned robot navigation in unknown terrains. Computer **22**(6), 37–43 (1989)
20. Sugihara, K.: Approximation of generalized voronoi diagrams by ordinary voronoi diagrams. CVGIP: Graph. Models Image Process. **55**(6), 522–531 (1993)
21. Tzionas, P.G., Thanailakis, A., Tsalides, P.G.: Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. IEEE Trans. Robot. Autom. **13**(2), 237–250 (1997)
22. Wang, C., Soh, Y., Wang, H., Wang, H.: A hierarchical genetic algorithm for path planning in a static environment with obstacles. IEEE Can. Conf. Electr. Comput. Eng. **3**, 1652–1657 (2002)
23. Jahanbin, M., Fallside, F.: Path planning using a wave simulation technique in the configuration space. In: Gero, J.S. (ed.) Artificial Intelligence in Engineering: Robotics and Processes. Computational Mechanics Publications, Southampton (1988)

24. Zelinsky, A.: Using path transforms to guide the search for findpath in 2D. Int. J. Robot. Res. **13**(4), 315–325 (1994)
25. Ioannidis, K., Sirakoulis, G.C., Andreadis, I.: A path planning method based on cellular automata for cooperative robots. Appl. Artif. Intell. **25**(8), 721–745 (2011)
26. Kostavelis, I., Boukas, E., Nalpantidis, L., Gasteratos, A.: Path tracing on polar depth maps for robot navigation. In: Cellular Automata for Research and Industry, pp. 395–404 (2012)
27. Chatzichristofis, S.A., Mitzias, D.A., Sirakoulis, G.C., Boutalis, Y.S.: A novel cellular automata based technique for visual multimedia content encryption. Optics Commun. **283**(21), 4250–4260 (2010)
28. Maeda, K.I., Sakama, C.: Identifying cellular automata rules. J. Cell. Automata **2**(1), 1–20 (2007)
29. Gray, L.: A mathematician looks at wolfram's new kind of science. Notices-Am. Math. Soc. **50**(2), 200–211 (2003)
30. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
31. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A*. J. ACM **32**(3), 505–536 (1985)
32. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading (1984)
33. Nalpantidis, L., Sirakoulis, G., Gasteratos, A.: Review of stereo vision algorithms: from software to hardware. Int. J. Optomechatron. **2**(4), 435–462 (2008)
34. Castaneda, V., Mateus, D., Navab, N.: Stereo time-of-flight. In: International Conference on Computer Vision (2011)
35. Khoshelham, K., Elberink, S.: Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors **12**(2), 1437–1454 (2012)
36. Nalpantidis, L., Kostavelis, I., Gasteratos, A.: Stereovision-based algorithm for obstacle avoidance. Intelligent Robotics and Applications, pp. 195–204 (2009)
37. Kostavelis, I., Nalpantidis, L., Gasteratos, A.: Supervised traversability learning for robot navigation. In: TAROS 2011, pp. 289–298. Springer (2011)
38. Labayrade, R., Aubert, D., Tarel, J.: Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation. In: Intelligent Vehicle Symposium, 2002, IEEE, vol. 2, pp. 646–651. IEEE (2002)
39. De Cubber, G., Doroftei, D., Nalpantidis, L., Sirakoulis, G., Gasteratos, A.: Stereo-based terrain traversability analysis for robot navigation. In: IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance, Brussels, Belgium (2009)
40. Huang, H., Guo, R.: Static floor field and exit choice for pedestrian evacuation in rooms with internal obstacles and multiple exits. Phys. Rev. E **78**(2), 021131 (2008)
41. Zheng, Y., Jia, B., Li, X.G., Zhu, N.: Evacuation dynamics with fire spreading based on cellular automaton. Physica A: Stat. Mech. Appl. **390**(1819), 3147–3156 (2011)
42. Procopio, M.J.: Hand-labeled DARPA LAGR datasets. http://ml.cs.colorado.edu/procopio/labeledlagrdata/ (2007)

Chapter 9
Cellular Robotic Ants Synergy Coordination for Path Planning

Konstantinos Ioannidis, Georgios Ch. Sirakoulis and Ioannis Andreadis

Abstract In this chapter, a unified architecture is proposed for a robot team in order to accomplish several tasks based on the application of an enhanced Cellular Automata (CA) path planner. The presented path planner can produce adequate collision-free pathways with minimum hardware resources and low complexity levels. During the course of a robot team to its final destination, dynamic obstacles are detected and avoided in real time as well as coordinated movements are executed by applying cooperations in order to maintain the team's initial formation. The inherit parallelism and simplicity of CA result in a path planner that requires low computational resources and thus, its implementation in miniature robots is straightforward. Cooperations are limited to a minimum so that further resource reduction can be achieved. For this purpose, the basic fundamentals of another artificial intelligence method, namely Ant Colonies Optimization (ACO) technique, were applied. The entire robot team is divided into equally numbered subgroups and an ACO algorithm is applied to reduce the complexity. As each robot moves towards to its final position, it creates a trail of an evaporated substance, called "pheromone". The "pheromone" and its quantity are detected by the following robots and thus, every robot is absolved by the necessity of continuous communication with its neighbors. The total complexity of the presented architecture results to a possible implementation using a team of miniature robots where all available resources are exploited.

K. Ioannidis (✉) · G.Ch. Sirakoulis · I. Andreadis
Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, GR, Greece
e-mail: kioannid@ee.duth.gr

G.Ch. Sirakoulis
e-mail: gsirak@ee.duth.gr

I. Andreadis
e-mail: iandread@ee.duth.gr

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_9

9.1 Introduction

Robot navigation is perhaps the most significant and active research field on the area of mobile robotics due to its applicability to a wide variety of tasks such as industry and human-supported works. The development of a rapid and efficient procedure which deals with the path planning problem is considered to be a key step for the motion of a mobile robot [68]. Path planning typically refers to the design of geometric specifications of the positions and orientations of robots in the presence of obstacles. An efficient path planner aims for the creation of a continuous motion that connects a starting point and a goal point in the configuration area of a mobile robot with the presence of obstacles (Fig. 9.1). The complexity of the problem is significantly increased when the developed framework is required to create collision free trajectories for every robot of a cooperative team. Cooperative robotic teams are extensively used in systems for accomplishing tasks where single robots fail to complete successfully their goals [46]. A variety of practical and potential applications can benefit from the use of a cooperative robot team, such as exploration of unknown areas [10], search and rescue [53] and formation control [25].

In general, the most widely known classification for path planning solution algorithms relies on the discrimination of the environments between static and dynamic. In the static case, all necessary information relative to the position of obstacles is known a priori while in dynamic planning, robots display a complete unawareness of their configuration area. For example, Tzionas et al. proposed in [63] an approach based on a retraction of free space onto the Voronoi diagram, which is constructed through the time evolution of Cellular Automata (CA) after an initial phase, during which the boundaries of obstacles are identified and coded with respect to their orientation. Despite its rapid execution, the approach suffers from a constraint of the

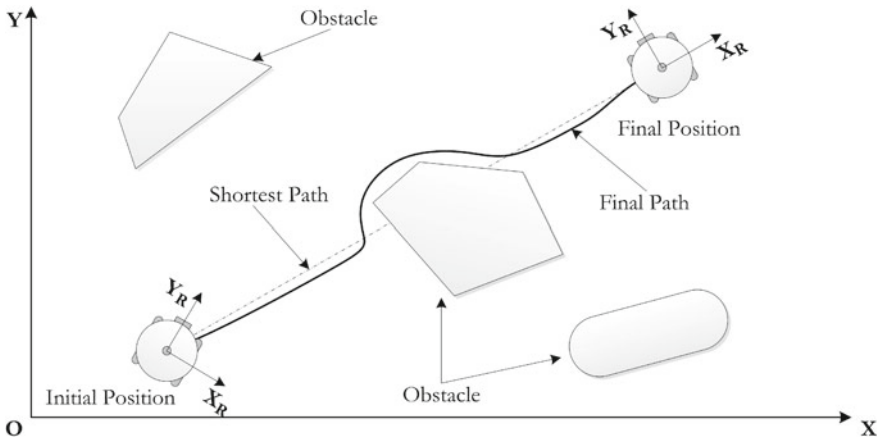


Fig. 9.1 An example of path planning that connects a starting point and a goal point in the configuration area of a mobile robot with the presence of obstacles

workspace, which is restricted in the sense that a generally shaped robot is enclosed within a diamond-shaped figure. A similar CA-based technique was proposed by Marchese in [47] where an anisotropic propagation of attracting potential values in a 4-D space-time using a multilayered cellular automaton (MCA) architecture was exploited. The algorithm executes a search for all the optimal collision-free trajectories following the minimum valley of a potential hypersurface embedded in a five dimensional space. However, efficiency cannot be obtained due to the uncompleted priority planning and thus, collisions between robots may occur. A cell based method was also proposed in [70] where a known environment was assumed. The configuration area was represented by occupancy grids and it was separated into a grid of equally spaced cells. The problem is converted to a graph-search problem and so, an A-star algorithm was applied [17]. Due to its features, the A-star approach increases the complexity of the system and thus, more processing resources are required in a real system. Dubins' theorem was also exploited for increased efficiency in [66] in order to deal with static environments. A genetic algorithm as well as a hierarchical structure of chromosomes to denote a possible path were used to identify the most optimal route. The method requires significant amounts of time rendering its implementation in real-time systems constrained.

Despite their efficiency, static environment approaches restrict their applicability in non-real world scenarios since most of the environments include dynamic obstacles. Path planners for dynamic environments may properly modify the path of a robot in cases of unexpected changes of its immediate environment. For example, the approach in [5] extracts a path from a static environment for a robot which is equipped with proximity sensors. Based on the sensors' readings, the robot bypasses possible obstacles that unexpectedly may block its route. The method requires part of information related to the status of the environment in order to be functional. A similar approach was proposed in [55] where a mobile robot modifies its path in the presence of semi-dynamic obstacles. The navigational planning is achieved with the application of a genetic algorithm until the robot reaches its final destination. In addition, a fuzzy-logic sensor fusion system was developed in [61] for target recognition, wherein the proposed path planning solver is based on a grid-map-oriented system that permits path revision through interactions with dynamical environments. An efficient dynamic system algorithm, using a neural dynamic model and a distance transform model was proposed in [67]. The approach develops efficient collision-free trajectories for a robot, and although dynamic programming is employed, computational cost remains in high levels. The A-star approach was also used in [24] as a global path planner to produce a series of sub-goal points to the target point. A potential field method was embedded as a local path planner in order to smooth the path between the preplanned sub-goal points. Thus, the method fails to cope with a real-time robot systems since global information of the entire configuration area is required. In order to produce smooth paths and reduce the complexity of the solution, a series of waypoints are interpolated in [2] with the use of a B-spline which is altered only in the area local to the obstacle. B-splines are piecewise series of polynomials and, therefore, the solution remains complex despite their low order. Finally, in [71], an improved Hopfield type neural network model was applied in

order to propagate the target activity among neurons, in the manner of physical heat conduction. The motion of a robot is defined through the dynamic neural network activity. In general, the method creates efficient paths for a robot, nonetheless, the exploitation of the neural network leads to high computational burden.

The discrimination of the path planning algorithms into either static or dynamic provides a base categorization without, nonetheless, any reference to the solution itself. A more proper differentiation is relied on the nature of the problem and the manner that an algorithm approaches its solution. Thus, research in path planning could also be classified into four basic categories: visibility graphs, potential fields, cell decomposition and heuristic algorithms [63]. Visibility graphs include the determination of a line collection in free space such that a connection of features of an object to those of another is accomplished [42]. A visibility graph can produce $O(n^2)$ edges for (n) features and so it is characterized by high complexity. On the contrary, potential field approaches exploit potential functions that are developed for obstacle avoidance in static environments. These approaches treat a robot's configuration as a point in a potential field that combines attraction to the goal and repulsion from obstacles. For example, the method proposed in [3] relies on the generation of a potential field by sigmoid and normal functions which eventually create the vector fields that control the velocity and heading of robot swarms. Similarly, relative headings to the goal and to obstacles, the distance to the goal and the angular width of obstacles were used in [36] to compute a potential field over the robot heading. The computed potential field controls the angular acceleration of the robot, steering towards the goal and far from the obstacles. In general, potential field approaches produce efficient trajectories however; higher accuracy can only be achieved by using higher order potentials, increasing the computational complexity. Heuristic approaches have been proposed as alternative solutions in order to simplify the problem. Ferguson proposed in [20] an extended version of a D-star algorithm [60] by using linear interpolation during each vertex expansion. Heuristic methods are characterized by their high time complexity which depends on the applied heuristic. On the contrary, cell decomposition methods were proposed to reduce the total complexity of the problem. Their main concept includes partitioning the free space into regions and identifying possible contacts between a single robot and obstacles in each region. For example, in [16], a variant of the A-star algorithm, namely Theta-star, propagates information along grid edges without constraining the paths to grid edges. As A-star based algorithms, the method requires a complete knowledge of the configuration area to produce the desired paths, thus, its implementation to a real system would be intricate. Nevertheless, Charalampous et al. in [12] combined the A-star with CA, and tested successfully the resulting method in real world planar environments. More specifically, the finite properties of the A-star algorithm were amalgamated with the CA rules to built up a substantial search strategy [11]. The corresponding algorithm's main attribute is that it expands the map state space with respect to time using adaptive time intervals to predict the potential expansion of obstacles.

Additionally, during the past few years, many artificial intelligence algorithms were employed as possible solutions to the problem so that the main drawbacks of the above solutions could be eliminated. In [43], a fuzzy based approach was proposed

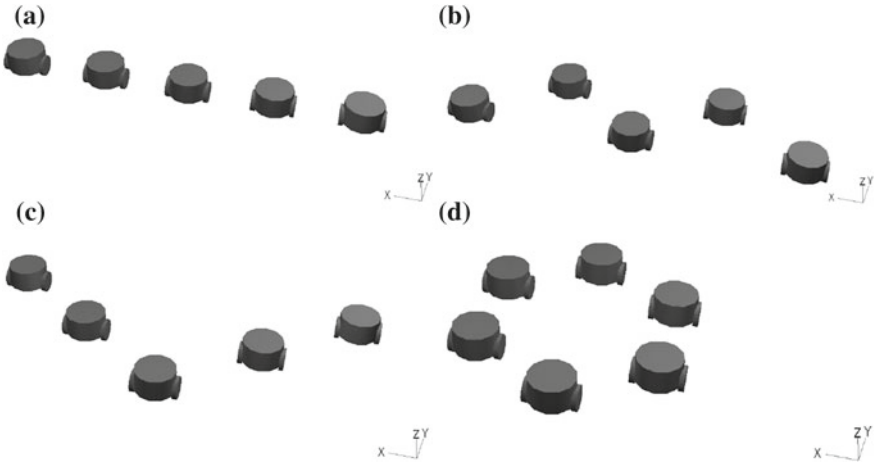


Fig. 9.2 Several formations of cooperative robotic teams

to navigate a mobile robot in an unknown dynamic environment filled with obstacles. The method requires for the robot to be equipped with a variety and highly accurate sensors to produce the desired pathways increasing the total cost of the system. More efficient paths were produced by the neural network in [69] nonetheless; in static environments and with high computational cost. A cell decomposition method along with an artificial intelligence method was used as a possible solution to the problem in [48]. Multilayered Cellular Automata (MCA) were applied where the configuration area is presented as a lattice of cells and four layers of identical grids are exploited for solving the path planning problem.

The majority of the above approaches deals with systems which include one robot and so only a single path should be extracted. The complexity of the solution is significantly increased in systems of multiple robots and is proportional to the total number of its members (Fig. 9.2). Such systems must complete further goals beyond the coverage of the desired distance. In case of a cooperative robot team, tracing paths for every robot becomes even more complex. The coordinated motion of a robotic team comprises one of the most widely studied research field in cooperative robotics and is known as formation control. Fredslund et al. in [26] achieved a collective behavior in a group of distributed robots using local sensing and minimal communication. Each robot references itself locally to one neighboring robot and keeps a certain bearing and distance by using an appropriate sensor. Moreover, a novel approach was proposed in [30] where formation structures are represented in terms of queues and formation vertices and formation control is accomplished using artificial potential trenches. A feedback law using Lyapunov type analysis was also exploited for a single robot and so collision avoidance and tracking of a reference trajectory was achieved [50]. Then, the method extends the resulted pathway to the case of multiple non-holonomic robots. Finally, a coordinated control scheme based

on a leader-follower approach was proposed in [18] so that formation maneuvers could be achieved. First and second order sliding mode controllers were used for asymptotically stabilizing the vehicles to a time varying desired formation.

The aforementioned methods for formation control and collision avoidance display high levels of computational complexity cost, despite their efficiency in both trajectory accuracy and formation immutability. Thus, their implementation in real systems could be either exclusionary or restricted. A minority of these methods could be potential for possible real system implementation nonetheless; robots must be equipped with farfetched sensors and high technical specifications which eventually increases the total cost of the system. On the other hand, their implementation in low cost robots restrict the number of tasks that could be executed leading to single purpose systems. For example, most of the commercial robots are equipped with digital cameras which are unable to be exploited since all the computational resources are occupied by the path planner. A low computational cost path planning approach could provide spare resources that are beneficial for exploiting the digital cameras to accomplish further tasks. However, these cameras capture low resolution images since higher resolution cameras require more processing power to analyze and manipulate a digital image. Instead, low resolution images are frequently inappropriate to achieve specific goals. Low resolution images can lead to false results when they are used to accomplish tasks like panoramic images or simultaneous localization and mapping (SLAM). Consequently, spare processing amounts could be used to increase the captured images' resolution with the application of image resizing methods. Several commonly used interpolation methods have been used for resizing images, such as nearest neighbor [38], bilinear [38] and bicubic interpolation [39]. In addition, a quadratic image interpolation method was proposed in [52] with adequate visual results nonetheless; its high computational cost prohibits its implementation. The preservation of the edges of a low resolution image comprises a crucial task and thus, a fuzzy decision system was developed in [45] to classify all the pixels of the input image into sensitive and non-sensitive class. Bilinear interpolation was applied to the non-sensitive regions while a neural network was used to interpolate the sensitive regions along the edges of the images. In order to decrease the computational complexity and preserve the satisfying image results, an edge-oriented method was proposed in [13] where the processed image is discriminated in non-edge and edge areas and each region is processed by a different type of interpolation method. The method achieves real-time image enlargement, despite the two stages of processing, nevertheless, the classification of the areas depends on a predefined threshold. Finally, the method in [44] initially estimates local covariance coefficients from the low resolution image which are sequentially used to adapt the interpolation at a higher resolution based on the geometric duality between the low-resolution and the high-resolution covariance. Despite its visually accurate resulted images, the method displays high levels of computational cost and thus, its application in real-time systems is restricted.

In this chapter, a unified architecture is presented for a robot team in order to accomplish several tasks and includes the application of an enhanced CA path planner combined with Ant Colonies Optimization (ACO) technique resulting to "Cellular

Robotic Ants". The presented path planner can produce adequate collision-free pathways with minimum hardware resources and low complexity levels. During the course of a robot team to its final destination, dynamic obstacles are detected and avoided in real time as well as coordinated movements are executed by applying cooperations in order to maintain the team's initial formation. The inherit parallelism and simplicity of CA result in a path planner that requires low computational resources and thus, its implementation in miniature robots is straightforward. The significant cooperations are limited to a minimum so that further resource reduction can be achieved. For this purpose, the basic fundamentals of ACO technique were applied. The entire robot team is divided into equally numbered subgroups and an ACO algorithm is applied to reduce the complexity. As each robot moves towards to its final position, it creates a trail of an evaporated substance, called "pheromone" . The "pheromone" and its quantity are detected by the following robots and thus, every robot is absolved by the necessity of continuous communication with its neighbors. The total complexity of the presented architecture results to a possible implementation using a team of miniature robots where all available resources are exploited. More specifically, the method was implemented in a cooperative robot team using a 3-D simulator, called Webots [51]. For testing purposes, the under study robot team was constituted of two subgroups with five robots each. All essential sensors, that all robots must be equipped, and their direct relations with the cell length and pheromone were introduced. The accuracy of the method was tested by using two different types of objects, rectangular and circular shaped. In both cases, the method created successfully collision free paths and the corresponding results exhibit the effectiveness and the robustness of the method.

The rest of the chapter is organized as follows. All the theoretical background for the CA and ACO algorithms is presented in Sect. 9.2 while the path planner, derived from the combination of CA and ACO, is presented in Sect. 9.3. Simulation results and the implementation of the architecture are presented in Sect. 9.4. Finally, conclusions are drawn in Sect. 9.5.

9.2 Cellular Automata and Ant Colony Optimization Principles

Cellular Automata (CA) were originally introduced by John Von Neumann [65] and his colleague Stanislaw Ulam [64]. CA can be considered as dynamical systems in which space and time are discrete and interactions are local. In general, a CA is consisted of a large number of identical entities with local connectivity arranged on a regular array. A finite Cellular Automaton could be defined by the quadruple:

$$\{d, q, N, F\} \tag{9.1}$$

From Eq. 9.1, variable d is a vector of two elements, m and n, denoting the vertical and horizontal CA dimensions, respectively. Both of these variables are expressed in number of cells. At each time step, the state of each cell is updated using a value

from the set $Q = 1, 2, \ldots, q-1$, called set of states. The neighborhood of each cell is defined by the variable N. For a 2-D CA, two neighborhoods are often considered, Von Neumann and Moore neighborhood. Von Neumann neighborhood is a diamond shaped neighborhood and can be used to define a set of cells surrounding a given cell $(x0, y0)$. Equation 9.2 defines the Von Neumann neighborhood of range r.

$$N^{v}_{(x0,y0)} = (x, y) : |x - x0| + |y - y0| \leq r \tag{9.2}$$

For a given cell $(x0, y0)$ and range r, Moore neighborhood can be defined by the following equation:

$$N^{M}_{(x0,y0)} = (x, y) : |x - x0| \leq r, |y - y0| \leq r \tag{9.3}$$

The transition rule f determines the way in which each cell of the CA is updated. The state of each cell is affected by the cell values in its neighborhood and its value on the previous time step, according to the transition rule or a set of rules. The state of every cell is updated simultaneously in the CA, thus, providing an inherent parallel system.

Consider for example a 2-D CA with two possible states, "0" and "1", as it is presented in Fig. 9.3. For this example, von Neumann neighborhood was used. In Fig. 9.3a, the central cell is stated as "1" at time t (black cell) while in Fig. 9.1b it is stated as "0" (white cell). The central cell updates its state to the next time step according to a simple XOR transition rule. If a central cell or any of its adjacent cells have the "1" state, at the next time step its state will be stayed into "1", as it is presented in Fig. 9.3, while its adjacent cells will update their state as "1".

CA have sufficient expressive dynamics to represent phenomena of arbitrary complexity and at the same time can be simulated exactly by digital computers, because of their intrinsic discreteness, i.e. the topology of the simulated object is reproduced in the simulating device. The CA approach is consistent with the modern notion of

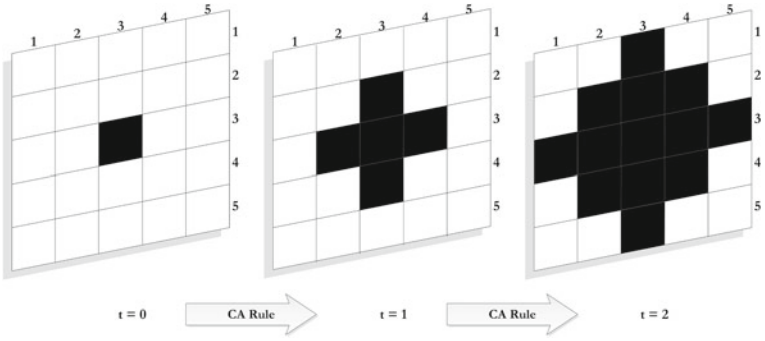


Fig. 9.3 2-D CA example with two possible states, "0" and "1". *Black cells* represent cells with state "1" while *white cells* represent "0" stated cells

unified space time. In computer science, space corresponds to memory and time to processing unit. In CA, memory (CA cell state) and processing unit (CA local rule) are inseparably related to a CA cell. Furthermore, CA are an alternative to partial differential equations [54, 62] and they can easily handle complicated boundary and initial conditions, inhomogeneities and anisotropies [31, 56].

The basic element of ACO algorithms is "ants" that is, agents with very simple capabilities which, to some extent, mimic the behavior of real ants [22]. Real ants are in some ways much unsophisticated insects. Their memory is very limited and they exhibit individual behavior that appears to have a large random component. However, acting as a collective, ants collaborate to achieve a variety of complicated tasks with great reliability and consistency [19], such as defining the shortest pathway, among a set of alternative paths, from their nests to a food source [4]. This type of social behavior is based on a common feature with CA, called self-organization, a set of dynamical mechanisms ensuring that the global aim of the system could be achieved through low level interactions between its elements [32]. The most vital feature of this interaction is that only local information is required. There are two ways of information transfer between ants: a direct communication (mandibular, antennation, chemical or visual contact, etc.) and an indirect communication, which is called stigmergy (as defined by Grassé [33]) and is biologically realized through pheromones, a special secretory chemical that is deposited, in many ant species, as trail by individual ants when they move [6]. More specifically, due to the fact that ants can detect pheromone, when choosing their way, they tend to choose paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source. This behavior is known as "auto catalytic" behaviour or the positive feedback mechanism in which reinforcement of the previously most followed route, is more desirable for future search. In ACO algorithms, an ant will move from point i to point j with probability:

$$\rho_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})} \tag{9.4}$$

where, $\tau_{i,j}^{\alpha}$ and $\eta_{i,j}^{\beta}$ are the pheromone value and the heuristic value associated with an available solution route, respectively. Furthermore, α and β are positive real parameters whose values determine the relative importance of pheromone versus heuristic information.

During their search for food, all ants deposit on the ground a small quantity of specific pheromone type. As soon as an ant discovers a food source, it evaluates the quantity and the quality of the food and carries some to the nest on their back. During the return trip, every ant with food leaves on the ground a different type of pheromone of specific quantity, according to the quality and quantity of the food.

In ACO algorithms, pheromone is updated according to the equation:

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{9.5}$$

where, $\tau_{i,j}$ is the amount of pheromone on a given position (i, j), ρ is the rate of the pheromone evaporation and $\Delta\tau_{i,j}$ is the amount of pheromone deposited, typically given by:

$$\Delta\tau_{i,j} = \begin{cases} 1/L_k, & \text{if ant } k \text{ travels on edge } i, j \\ 0, & \text{otherwise} \end{cases} \tag{9.6}$$

where L_k is the cost of the kth tour of an ant (typically is measured as length). Finally, the created pheromone trails will guide other ants to the food source.

Consider for example the experimental setting shown in Fig. 9.4. The ants move along the path from food source F to the nest N. At point B, all ants walking to the nest must decide whether to continue their path from point C or from point H (Fig. 9.2a). A higher quantity of pheromone on the path through point C provides an ant a stronger motivation and thus a higher probability to follow this path. As no pheromone was deposited previously at point B, the first ant reaching point B has the same probability to go through either point C or point H. The first ant following the path BCD will reach point D earlier than the first ant which followed path BHD, due to its shorter length. The result is that an ant returning from N to D will trace a stronger trail on path DCB, caused by the half of all the ants that by chance followed path DCBF and by the already arrived ones coming via BCD. Therefore, they will prefer path DCB to path DHB. Consequently, the number of ants following this path

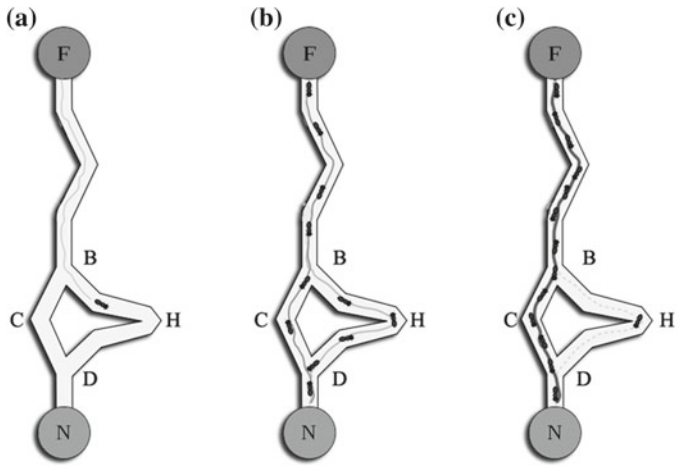


Fig. 9.4 An example of real ants colony: **a** An ant follows BHD path by chance, **b** Both paths are followed with same probability and **c** Larger number of ants follow the shorter path

will be increased during time than the number of ants following BHD. This causes the quantity of pheromone on the shorter path to grow faster than the corresponding longer one. Consequently, the probability with which any single ant chooses the path to follow is quickly biased towards the shorter one.

The ACO algorithms are basically a colony of artificial ants or cooperative agents, designed to solve combinatorial optimization problems. These algorithms are probabilistic in nature because they avoid the local minima entrapment and provide very good solutions close to the natural solution [7]. ACO algorithms are extensively used to a variety of applications such as the travel salesman problem [23], image retrieval [40, 41], classification [49], electrical load pattern grouping [14], video games [57], seismic methods [15], communications networks [21], etc. Moreover, as it is proposed in this paper, ACO algorithms were also used for solving the path planning in a team of robots and most of the effort was to implement the algorithm in real systems. More specifically, in [58] heat applicators and sensors were used as virtual pheromone and detectors, respectively, so that ACO algorithm could be functional. Furthermore, Garnier et al. in [29] used a visual system, a computer and a projector to track robots, process the data and project the light, respectively. Moreover, Garnier et al. in [27] proposed a group of ant-like robots that had to establish a route between a starting area and a target area in a network of corridors, mimicking the experiments we performed with ants in authors' previous studies [28]. For technical convenience pheromone trails were replaced by light trails projected along the paths followed by the robots by a video projector (as proposed in [58] and implemented in [29]). Robots could detect and follow these light trails thanks to two photoreceptors that mimic the antennae of the ants as also described in the corresponding review article of Akst [1]. Finally, in [34, 35], RFID stickers were used as data carriers, which were a priori placed in the area, and they stored data representing the pheromone levels. At each robot of a team, an RFID reader/writer was mounted in order to read/write the data stored in the RFID stickers.

9.3 Cellular Ants: A Combination of CA and ACO Algorithms for Path Planning

One of the main goals of the proposed method is to create collision free trajectories for every robot of a cooperative team. No a priori knowledge of the configuration area is required. Obstacle avoidance must be achieved in real time. Knowing their final position, that is the end of a straight line path, robots can move randomly in the configuration area, according to ACO algorithm. To prevent a scattered formation due to either an obstacle or a complete absence of pheromone, cooperations between the members of the team are applied so that their formation could be regained or retained immutable, respectively. According to the ACO algorithm, every single ant is governed by a set of simple behavior rules, leading to an uncomplicated approach of the path planning problem. Due to CAs, these behavior rules are applied simultaneously

to all ants, in a discrete and iterative way. A concurrent evolution of the entire system ensures the rapid formation of all possible trajectories. Furthermore, the proposed method covers the need for self-organization, since the utilized artificial intelligence algorithms embody this particular attribute. In the following subsections, the proposed method is described in detail. Simulation results are provided as well.

9.3.1 Proposed Method

First, an appropriate CA must be defined, meaning that the variables in Eq. 9.1 must be specified. The configuration area, in which the robot team operates, is considered to be a 2-D lattice and so is divided into a simple rectangular grid of identical square cells. The dimensions of the CA (variable d) are selected based on the dimensions of the configuration area and the cell length. Let $m \times n$ be the dimensions of the CA, therefore $d = m, n$, and w the length of each cell.

Moreover, each cell can take a finite number of possible states, named set of states $Q = 0, 1, 2, \ldots, q$. Variable q is proportional to the number of robots that comprise the team and the predefined pheromone levels. Firstly, vector $F = 0$ represents the state of a cell unoccupied by a robot, an obstacle or pheromone, called a free cell. In addition, vector $R = 1, 2, \ldots, r$ represents all possible states of a cell occupied by a robot. Due to the duality of their nature as a CA cell and as an artificial ant, a cell occupied by a robot is labeled as a cellular ant. For reasons of complexity, the entire robot team is divided into smaller subgroups, equally numbered and forming the same pattern. Furthermore, $P = r + 1, r + 2, \ldots, \rho$ is a vector of all possible pheromone states, where ρ denotes a cell with the maximum pheromone level. Finally, variable $W = \rho + 1$ indicates a cell occupied by an obstacle. These vectors were created to avoid overlapping between possible states according to the following equations:

$$F \cup W \cup R \cup P = Q \;\&\; F \cap W \cap R \cap P = Q \tag{9.7}$$

Summarizing, every CA cell could obtain one possible state at each time step:

- Free cell: $cell(x, y) = 0$
- Cellular ant: $cell(x, y) = r$ where $v \in N : r_v \in R$
- Pheromone cell: $cell(x, y) = \rho$ where $v \in N : \rho_v \in R$
- Obstacle cell: $cell(x, y) = \rho + 1$ where ρ the state of maximum pheromone level

where x and y are the Cartesian coordinates of a cell in the CA grid.

Every CA cell evolves its state according to its current state and the state of each neighboring cell. For the proposed method, Moore neighborhood was used with range $r = 1$ (Eq. 9.3). Moreover, cells located on the frontier of the grid evolve their state using null boundary conditions, meaning that all cells of the firstlast row and those of the first/last column are enduringly in the 0 state.

At every time step, all cells update their state simultaneously by applying the appropriate local transition rule of the CA, $F : Q \leftrightarrow Q$, so that:

$$cell_{i,j}^{t+1} = F cell_{i,j}^{t}, cell_{i-1,j-1}^{t}, cell_{i-1,j}^{t}, \ldots, cell_{i+1,j+1}^{t} \tag{9.8}$$

Three different sets of CA rules were created for achieving different operations: collision avoidance, pheromone update and formation control.

9.3.1.1 CA Rules for Collision Avoidance

Appropriate CA rules were created so that objects could be avoided. As a cellular ant moves towards to its final position, it scans its Moore neighborhood at every time step in order to detect a potential obstacle. Due to this attribute, both static and dynamic objects can be detected. Depending on the status of its adjacent cells and its direction, the appropriate CA rule is selected for the evolution of the state of the corresponding cellular ant. Considering an obstacle detected by a cellular ant, at first, the applied CA rule is chosen according to the position of the cellular ant in the formation. If it is positioned on the right of the central cellular ant of the formation, the right pathway is selected to bypass the obstacle. On the contrary, the left pathway is selected in case of a left positioned cellular ant. Subsequently, the rest of the avoidance process is akin with a wall following behavior. If the obstacle is avoided and the cellular ant regain its position over the desired shortest path, suitable CA rules are applied in order to continue its way to its final destination. Table 9.1 illustrates a small subset of eight collision avoidance CA rules out of a total of forty possible CA rules. Figure 9.3 represents the application of CA rules for obstacle avoidance as a schema. More specifically, in Fig. 9.5a, a cellular ant detects that its NW and its N adjacent cells are occupied by obstacles, its NE cell as a free cell, "detects" that its SW is a cellular ant cell and its W is an obstacle cell. By applying the appropriate CA rules, at time $t + 1$, the under study central cellular ant cell is stated as a pheromone cell with maximum pheromone quantity and its NE free cell becomes a cellular ant

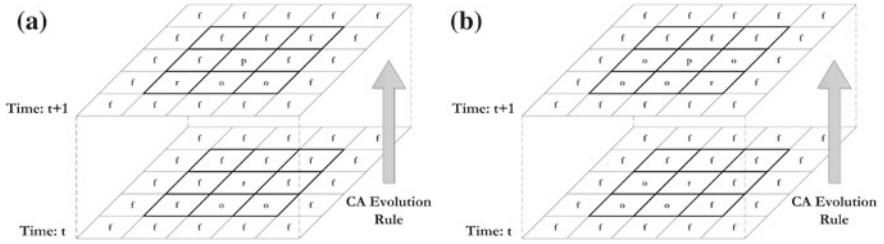


Fig. 9.5 Schematic presentation of two CA rules for collision avoidance: f denotes a free cell, r denotes a cellular ant cell, o an obstacle cell and ρ a pheromone cell with the highest possible pheromone state. **a** NW and N adjacent cells are obstacle cells and **b** N, NE and E adjacent cells are obstacle cells

Table 9.1 A subset of CA rules for collision avoidance

Cell state									
At time t									At time $t+1$
$S_{i,j}$	$S_{i-1,j-1}$	$S_{i-1,j}$	$S_{i,j+1}$	$S_{i,j+1}$	$S_{i+1,j+1}$	$S_{i+1,j}$	$S_{i+1,j-1}$	$S_{i,j-1}$	$S_{i,j}$
r	f	f	f	f	f	f	o	f	f
f	f	r	f	f	f	f	f	o	r
r	f	f	f	f	o	f	f	f	f
f	f	r	f	o	f	f	f	f	r
r	f	f	f	f	f	o	o	f	f
f	r	f	f	f	f	f	f	o	r
r	f	f	f	f	o	o	o	f	f
f	f	f	f	r	o	o	f	f	r

S denotes cell state, (i, j) Cartesian coordinates of cell on the CA, r a cellular ant cell, o an obstacle cell and f a free cell

cell. Essentially, a state transition is occurred between the central cell and its NE cell. Figure 9.5b illustrates a similar situation with different states of its adjacent cells.

9.3.1.2 CA Rules for Simulating the Pheromone Functions

To simulate all relative functions of pheromone, appropriate CA rules were also created. There is an immediate correlation between the state of a pheromone cell and the corresponding pheromone value. The state of a pheromone cell is relevant to the detected pheromone quantity (value $\tau^{a}_{i,j}$). Every cellular ant scans its front adjacent cells in order to be stated with the appropriate pheromone state. If a cell is marked with the highest pheromone state, it will be more likely for the cellular ant to follow this trail, according to Eq. 9.4. Moreover, supplementary CA rules were created in order to simulate the pheromone evaporation process. A pheromone cell surrounded by either free cells or pheromone cells updates its state according to Eqs. 9.5 and 9.6 by decreasing its value. Furthermore, the evaporation rate is expressed as state per time steps due to the quantized states of the CA. Depending on the pheromone levels and the evaporation rate, numerous CA rules could be used to simulate both the stigmergic behavior of a cellular ant and the evaporation of the pheromone. Table 9.2 illustrates a small subset of eight pheromone update CA rules, while Fig. 9.4 represents a schematic example of their application. In Fig. 9.6a, a cellular ant, namely r, detects two of its adjacent cell to be stated as pheromone cells, $\rho 5$ and $\rho 1$, respectively. The $\rho 5$ cell has a higher pheromone state meaning that it is occupied by a higher amount of pheromone. According to Eq. 9.4, it is more likely that the cellular ant will follow this path and thus, the appropriate CA rule is applied in order this pathway to be followed. Additionally, in Fig. 9.6b, a pheromone cell with state $\rho 5$ at time t updates its state by decreasing its state in order to simulate the pheromone evaporation.

Table 9.2 A subset of CA rules for collision avoidance

Cell state									
At time t									At time $t+1$
$S_{i,j}$	$S_{i-1,j-1}$	$S_{i-1,j}$	$S_{i,j+1}$	$S_{i,j+1}$	$S_{i+1,j+1}$	$S_{i+1,j}$	$S_{i+1,j-1}$	$S_{i,j-1}$	$S_{i,j}$
r	f	f	f	f	f	f	o	f	ρ
f	f	f	r	f	f	f	f	o	r
r	f	f	f	f	f	f	f	f	f
f	f	r	f	o	f	f	f	f	r
r	f	f	f	f	f	$\rho 2$	$\rho 5$	f	ρ
$\rho 2$	f	r	f	f	f	f	f	$\rho 5$	r
ρ	f	f	f	f	f	f	f	f	ρ
$\rho 1$	f	f	f	f	f	f	f	f	$\rho 2$

S denotes cell state, (i, j) Cartesian coordinates of cell on the CA, r a cellular ant cell, o an obstacle cell and f a free cell and ρ pheromone cells (increased pointers mean low levels of pheromone)

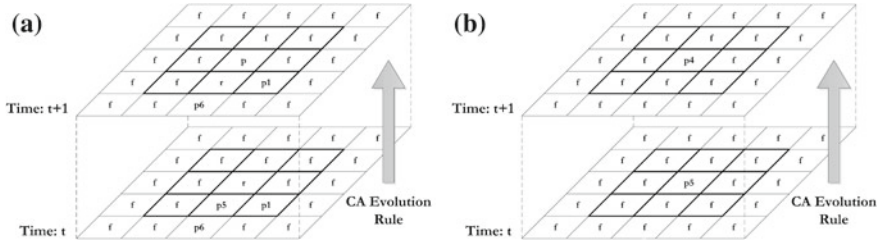


Fig. 9.6 Schematic presentations of two CA rules for simulating the pheromone functions: f denotes a free cell, r denotes a cellular ant cell and ρ a pheromone cell (indices represent pheromone level; higher index value corresponds to higher pheromone level, state p without any index represents a pheromone cell with the highest possible pheromone state). **a** Following the strongest pheromone trail and **b** Simulating the pheromone evaporation process

9.3.1.3 CA Rules for Formation Control

Each subgroup of cellular ants must retain and regain their initial formation during the entire route. If an obstacle is detected by a cellular ant of a subgroup, the formation of its subgroup will be scattered. Moreover, in case of complete absence of pheromone, according to Eq. 9.4, a cellular ant could move towards to any of its adjacent free cells with the same probability leading to a scattered formation as well. In order to prevent such behaviors, suitable CA rules were created based on cooperations between the cellular ants of every subgroup. It is assumed that every cellular ant has a substandard ability to communicate with its immediate neighbors. The coordinates of every cellular ant in the grid are required in order to complete the formation control process. Therefore, these coordinates are handled as the exchanged data. Depending on these data, all the necessary decisions are taken by each cellular ant meaning, whether to swap positions in the formation with its neighbors, move one cell to their destination or just stay still until the formation is regained by the other

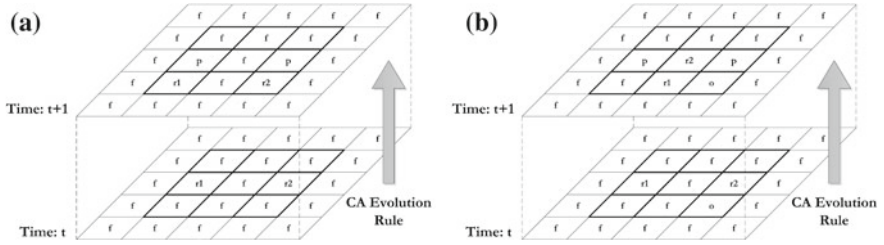


Fig. 9.7 Schematic presentations of two CA rules for formation control: f denotes a free cell, $r1$, $r2$ denote cellular ant cells, o an obstacle cell and ρ a pheromone cell with the highest pheromone state. **a** Moving to their goal and **b** Exchanging positions in formation

cellular ants of the subgroup. These decisions are expressed as simple CA rules in order to evolve their state to the next time step. Figure 9.7 illustrates two examples of the CA rules set for formation control. In Fig. 9.7a, no pheromone is detected by the $r1$ and $r2$ cellular ants and therefore, they communicate in order to proceed to data exchange. Since $r2$ finds its neighbor $r1$ to its vertical position and vice-versa, both cellular ants commonly decide to update their state in order to move one cell towards their destination. On the contrary, in Fig. 9.5b, cellular ant $r2$ detects an obstacle and therefore, it will try to bypass it leaving its position in the formation. At the same time, the $r1$ cellular ant discovers that it has smaller vertical distance than the predefined with $r2$ (after communication). Thus, they commonly decide to exchange their position in the formation in order to regain the formation as soon as possible.

For example, consider a cellular ant cell with state r and coordinates (i, j) with no obstacle found in its neighborhood at time t. At the given time, let us assume that all cells of its Moore neighborhood have the following states: $cell^t_{i,j} = a0 = r$, $cell^t_{i-1,j-1} = a1 = f$, $cell^t_{i-1,j} = a2 = f$, $cell^t_{i-1,j+1} = a3 = f$, $cell^t_{i,j+1} = a4 = f$, $cell^t_{i+1,j+1} = a5 = f$, $cell^t_{i+1,j} = a6 = f$, $cell^t_{i+1,j-1} = a7 = f$, $cell^t_{i,j-1} = a8 = f$. When applied to Eq. 9.8, these values result to $cell^{t+1}_{i,j} = a0^{t+1} = F(a0^t, a1^t, a2^t, a3^t, a4^t, a5^t, a6^t, a7^t, a8^t) = F(r, f, f, f, f, f, f, f, f) = r$. Furthermore, at time step t, the free cell with coordinates $(i + 1, j)$ will have the following values $a0 = f$, $a1 = f$, $a2 = r$, $a3 = f$, $a4 = f$, $a5 = f$, $a6 = f$, $a7 = f$, and $a8 = f$. Applying simultaneously the appropriate CA rule, the corresponding results are: $cell^{t+1}_{i,j} = a0^{t+1} = F(a0^t, a1^t, a2^t, a3^t, a4^t, a5^t, a6^t, a7^t, a8^t) = F(f, f, r, f, f, f, f, f, f) = r$. In other words, with the use of simple rules, the cellular ant cell manages to cover the distance of a cell to reach the goal position.

The main aims of the method are both the creation of collision free trajectories in dynamic environments and a fixed formation for every subgroup of the team. In case of a scattered formation, the members of the subgroup must recover the formation as soon as possible with the minimum requirements. For reasons of simplicity and for the given example, a straight line formation was applied. All cellular ants of the first

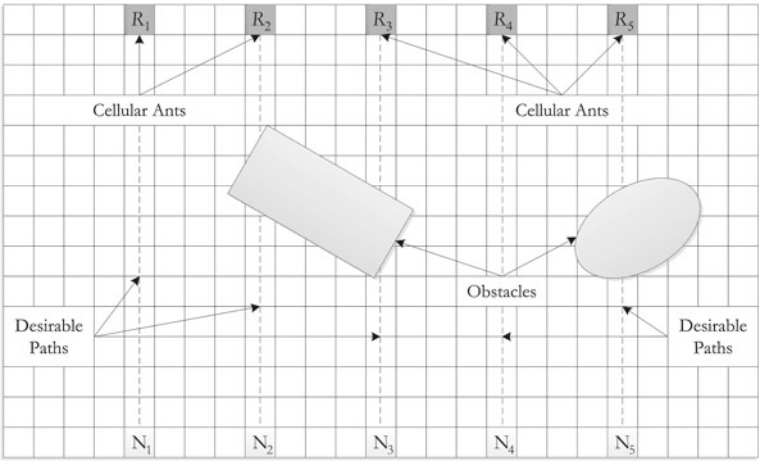


Fig. 9.8 Initial state of the CA grid, where *R* denotes cellular ant cells and *N* final positions (nests)

subgroup are deployed in either the first row or the first column of the CA having a specific vertical or horizontal distance, respectively. The forthcoming subgroups are subsequently introduced at the same positions in the grid after a user defined time period. Figure 9.8 illustrates the initial state of the CA grid for the aforementioned example using a first row deployment of the cellular ants.

For this particular example and for testing purposes, this specific type of formation was selected due to the nature of the pattern that cellular ants form [37]. Consider a real system consisting of multiple robots, each equipped with a digital camera. Images taken from each camera have a horizontal difference presenting the same scenery. Many image processing algorithms exploit this difference for a variety of applications, such as resolution enhancement [59] and panoramic images [9].

The method assumes that the initial and the final position of every cellular ant are its food source and its nest, respectively. In case of a position exchange in the formation, the corresponding cellular ants will exchange their final positions-nests, as well. At every time step, each cellular ant scans its Moore neighborhood in order to detect any possible obstacle. If an object is detected, the appropriate CA rule for obstacle avoidance is selected and applied without requiring any further process, e.g. pheromone detection. Otherwise, every cellular ant will scan its front adjacent cells in order to detect any pheromone quantity. The probability for every front cell is calculated according to Eq. 9.4 and a quantization is applied. Each quantized probability is mapped to a pheromone state and every cell is marked with the corresponding pheromone state. A higher pheromone quantity means that the specific path is more likely to be followed. Moreover, the formation of the subgroup is scattered either when an obstacle is detected or in cases of complete pheromone absence. In order to keep the formation of the subgroup immutable (lack of pheromone) or regain it (obstacle detection), formation control principles are applied where all decisions are

expressed as CA rules. These functions are achieved through cooperations between the cellular ants of a subgroup, meaning that data relative their position in the grid are exchanged. Every cellular ant knows about the presence of its two nearest cellular ants with which data exchange is achieved. In case of a deviated cellular ant from its desired shortest path due to an obstacle, its absence can be detected by its neighbors using its coordinates. If a cellular ant bypasses an obstacle from the left pathway, the horizontal distance with its left neighbor will be decreased while the distance with its right neighboring cellular ant will be increased. Thus, a position exchange between the cellular ant and its left neighbor will occur while the right neighbor will wait until the exchange is completed. When swapping is completed, neighboring cellular ants synchronize their actions by exchanging all necessary data, until all vertical coordinates are equalized. All formation control functions are continuously applied till the formation is regained, searching only for obstacle cells since the method is applicable in dynamic environments. When the straight line formation is regained, the entire subgroup will move towards to its final position as a team, reaching their targets synchronized.

As a cellular ant moves to a free or a pheromone cell, its state will be updated with the corresponding CA rule and thus, at the next time step, it will be stated as a pheromone cell with the maximum defined pheromone state. The cellular ants of the next subgroup detect these pheromone trails and consequently, no formation control principles are required in order to retain or regain the formation of the next subgroup. This advantage reduces the complexity that the formation control process introduces. Additionally, in a real system, the formation control process requires the members of the team to be able to communicate and thus, more time and resources are needed in order to keep the formation fixed. By using simple sensors for pheromone detection, if some pheromone is detected, cooperations and data exchange are no longer necessary. Finally, according to ACO algorithm, past events or situations could be expressed as modifications or "unusual" status of pheromone, acting as a local memory, since it is deposited by the anterior ants. In the proposed method,

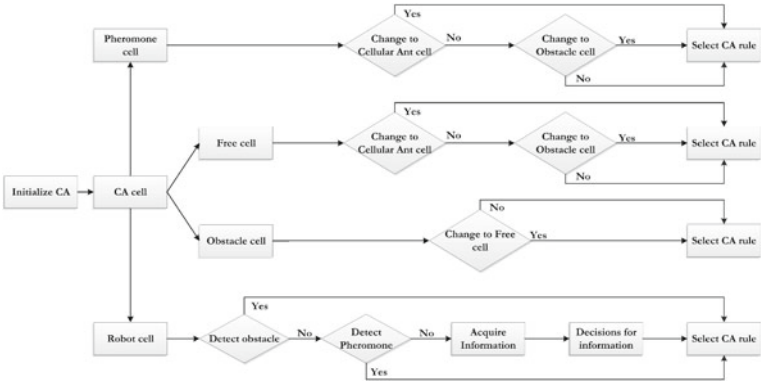


Fig. 9.9 Flowchart of the proposed method

these "unusual" pheromone allocations are expressed as a position exchange in the formation and are detected by the forthcoming cellular ants in order to prevent a false pheromone trail pathway. The method in a form of a flowchart is provided in Fig. 9.9.

9.4 Simulation Results

For testing purposes of the proposed method, a simulation environment was created and some results are illustrated in Fig. 9.10. The under study environment operates under Microsoft Windows OS and was created using the C# program language and a graphical user interface. The created simulator includes multiple subgroups constituted of numerous cellular ants. For testing purposes, each subgroup is comprised by five cellular ants and a straight line formation was applied. The program provides the ability to the user to define the dimensions of the CA, the number of cellular ants for all subgroups, the number of iterations-time steps and to draw rectangular or elliptical shapes as possible objects. Additionally, the pheromone levels and the evaporation ratio are also user defined. For the examples illustrated in Fig. 9.10, the dimensions of the CA are 21×25 cells with a number of time steps equal to 100. Every subgroup comprises by 5 cellular ants and is inserted in the grid after 25 time steps. Also, 10 pheromone levels were selected, meaning that 10 possible states can be used for the pheromone cells and an evaporation rate equal to 1 state per 10 time steps. Cellular ants are denoted with the letter R and corresponding numbers representing their position in the formation; object cells are depictured with darkened green squares, nests (final positions) with green and denoted with the letter N and finally, pheromone cells are illustrated with tones of blue depending on the quantity of the pheromone. As Fig. 9.10 shows, initially, the CA path planner creates collision free paths for every cellular ant using the formation control rules since no pheromone is deposited on the grid. At each time step, every cellular ant moves towards its final destination avoiding all possible obstacles found on its desired shortest route and leaving a pheromone trail on the grid. After a short period, the second subgroup leaves from their food source and thus, amplifying the existing pheromone trails and so on. Figure 9.10a illustrates the simulation results in case of rectangular objects while Fig. 9.10b presents the case of elliptical objects. For comparison reasons, all simulation parameters are kept immutable for both cases.

Simulation results indicate that the proposed method can produce accurate collision free trajectories for every cellular ant with low complexity since ACO principles are applied. Due to the fact that the proposed method mainly uses a CA, time complexity is found to be approximately $O(m \times n)$ as the majority of the CA algorithms, where $m \times n$ are the dimensions of the rectangular grid, meaning that time complexity is proportional to the CA dimensions. Moreover, time complexity is kept low in any environment, both static and dynamic. The procedure retains the principles of self organization of ACO algorithms and concurrently uses CA to achieve our goal. Furthermore, the whole process is accomplished without any interference of a central

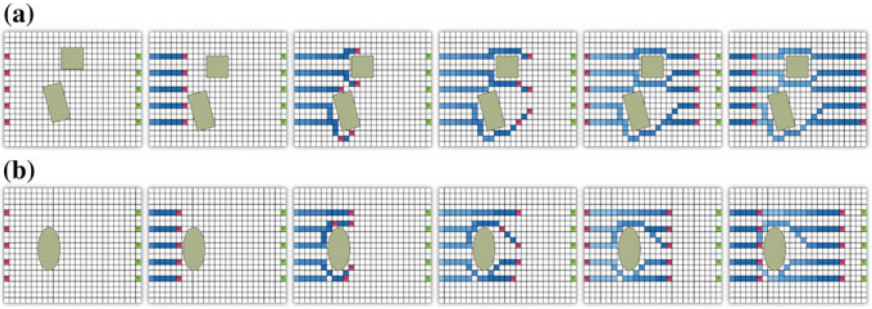


Fig. 9.10 Simulation results for **a** rectangular objects and **b** *elliptical* objects, where cellular ants are denoted with the letter *R* and a number (formation position), final positions are represented by *pale gray cells*, object cells are illustrated by *darkened cells* and pheromone cells are illustrated with tones of gray

control, making the system autonomous, and is completely parallel. Finally, due to the fact that each subgroup consists of multiple robots, a cellular ant could wrongly follow a pheromone trail leading to a scatter formation. Using the advantages of the CA path planner, this problem could be resolved as depicted in Fig. 9.10.

9.4.1 Implementation of the Method in a Simulated Cooperative Robot Team

The proposed method was implemented in a simulated cooperative robot team using a three dimensional simulator, called Webots [51]. All robots and instances simulated in Webots are actually real robots that either can be purchased or used for educational reasons. For our experiments, multiple miniature robots were used, called e-puck [8]. All robot controllers needed for the implemented method were created using the language C. E-puck robot is equipped with a variety of sensors which can be used for the implementation and its hardware structure is considered to be simple. To be fully functional, the method requires that every robot of the team must be equipped with specific types of sensors such as distance sensors, differential wheels, communication modules for data transmission/receipt and appropriate sensors for pheromone detection. Real e-puck robot is equipped with all the above sensors as well as the corresponding e-puck robot in Webots. A more detailed description about the sensors that were used for the implementation is presented below.

The proposed method requires every robot to be equipped with specific types of sensors so that it can have a completely awareness of its environment. Considering that, every robot could avoid possible obstacles, detect pheromone trails and

communicate with its neighboring robots during its route to its final destination. Consequently, for every action different type of sensors is needed.

9.4.1.1 Sensors for Obstacle Avoidance

E-puck robot uses infrareds as distance sensors to detect obstacles in the configuration area. For reasons of correspondence, infrared distance sensors were also used for the implementation in Webots. Unfortunately, as Fig. 9.11 illustrates, infrareds on the real e-puck robot are misplaced. Due to the quantization of the configuration area, obstacle detection cannot easily be achieved, leading to a less accurate system. For example, if an obstacle with small size is present between two infrared sensors, due to the distance between them, the robot will not be able to detect the object. By dividing the configuration area into smaller cells, the system can be more accurate at the expense of memory resources. However, in systems with miniature robots, memory restrictions are of great importance.

An obstacle can be detected only if it is in the range of a distance sensor so that the corresponding cell can be stated as an obstacle cell. Consequently, the cell length must be at least equal to the distance sensor range. Small memory amounts are required for large cells but less accurate readings from the sensors are achieved. Conversely, using small cell length, more accurate readings can be processed from the sensors but memory demands exceed the limit. To overpass this problem, the best solution is to define an average cell length, and use numerous sensors. For that reason, some modifications are required to be done on the e-puck robot. Different number of proximity sensors was attached on the robot to define the appropriate cell length so that maximum accuracy could be achieved. All distance sensors were

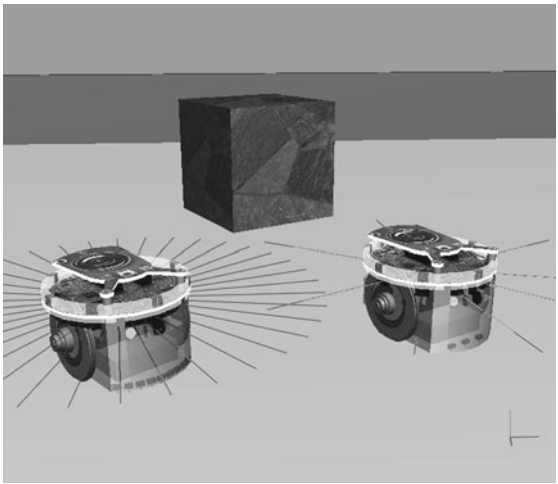


Fig. 9.11 From *left* to *right*: modified and real e-puck robot. The continuous lines represent the range of the infrared sensors

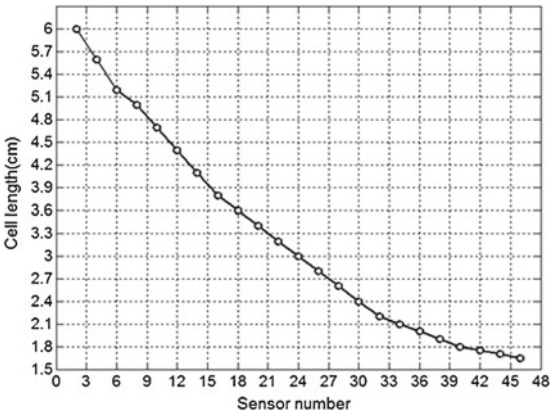


Fig. 9.12 Appropriate cell length for different number of proximity sensors

placed around the e-puck robot every time with different angle. Figure 9.12 illustrates the cell length of the system in regard to the required number of distance sensors so that an object could be detected at any angle.

In case of minimum sensor number, big cell length is required and therefore less memory is needed. Nevertheless, less accuracy is achieved and thus, some objects may not be detected. As the cell length is decreased, more sensors are required so that small obstacles can be detected. High accuracy is achieved but more memory is required. Therefore, to achieve maximum accuracy, multiple sensors must be used without using inefficient number of memory. For that reason, thirty six infrared sensors for obstacle avoidance were used instead of eight that real e-puck robot has. Each sensor was placed on the circumference of each robot having a distance of ten degrees with its neighboring sensors. Figure 9.11 illustrates the position of each sensor on a robot while Figs. 9.11 and 9.14 present the modified e-puck robot. Moreover, to state a free cell as an obstacle cell, the readings from the sensors must be compared with a threshold, relative to cell length. For accuracy reasons, this number is equal to the weighted sum of the sensors readings which are physically included in the correspondence cell. A 1×5 Gaussian convolution operator was used as coefficient. For example, in Fig. 9.13, assume that the value which describes the state of the north cell is equal to:

$$\begin{aligned} value = {} & 0.0545 \times a0 + 0.2242 \times a1 + 0.4026 \times a2 + 0.2242 \times a3 \\ & + 0.0545 \times a4 \end{aligned} \tag{9.9}$$

where $a0$, $a1$, $a2$, $a3$ and $a4$ are the readings from the sensors which are found in the north cell, from left to right. This value is compared with a threshold and if it is greater, the cell will be stated as obstacle cell.

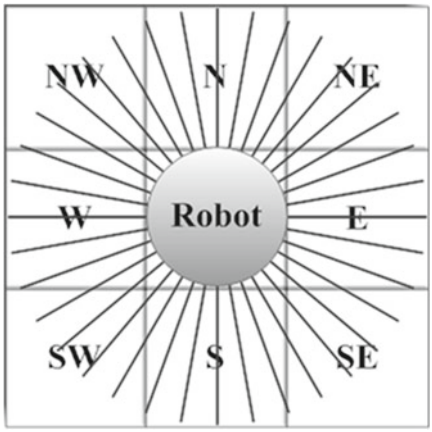


Fig. 9.13 Sensor Positions on a Robot

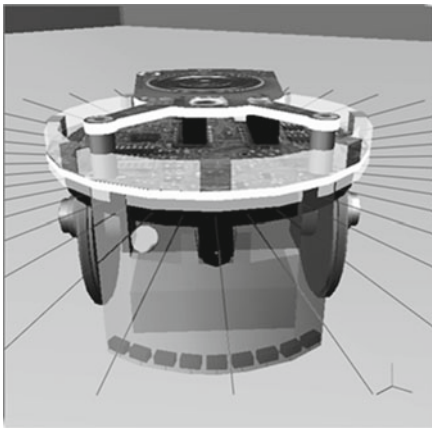


Fig. 9.14 Modified e-puck robot with infrared ground sensors

9.4.1.2 Sensors for Pheromone Detection

A special component, given by Webots and called pen, was attached to every robot. The pen component models a pen attached to a mobile robot, typically used to present the trajectory of the robot. The created trails can be considered as pheromone paths which can be detected by every robot. Essentially, the quantity of pheromone can be simulated by tones of gray created by the pen on the ground. A tone near to black represents a high accumulation of pheromone while near to white tones correspond to low levels of pheromone. A transition from black to white tones simulates the evaporation process of pheromone. Moreover, appropriate sensors must be selected so that the tones of gray can be detected. For this purpose, multiple infrared sensors were attached to every robot looking down to the ground, as illustrated in Fig. 9.14. According to their structure, infrareds can detect the amount of

the reflected beam which is transformed to electrical signal. If a sensor measures a trail with a black tone, a high value will be resulted. In case of closely white trails, small readings from the sensors will occur. As in obstacle avoidance, these readings are combined using a Gaussian convolution kernel. Finally, the resulted values are compared with thresholds to produce different pheromone levels for the pheromone cells of the CA.

At this point, it should be mentioned that in real e-puck robots, electrical circuits using infrared sensors can be attached on their front for the same purpose. Three infrared sensors are attached under the camera of the robot, facing the ground, and are used as ground sensors to detect possible chromatic trails.

9.4.1.3 Communication Modules

In case of absence of pheromone or a scattered formation, cooperations between the robots of each subgroup must be achieved so that their formation can be either retained or regained, respectively. Cooperation tasks are accomplished using data exchange through a communication link. Each robot communicates with its neighbors and swaps information about its position in the grid. Depending on these data, robots decide whether to exchange their position in the formation, move towards their final destination or remain to their current position. Real e-puck robot is equipped with a Bluetooth module for serial communications. In Webots, the modified e-puck robot uses a transmitter and a receiver component to achieve interactions. Essentially, these two components simulate the operations that a real serial communication module can complete. Each robot must scan its neighborhood to avoid obstacles, detect possible pheromone trails and communicate with other robots, if necessary, to update its state to the next time step. Thus, the required time to create its path is proportional to the execution time of each above operation. The total execution time is equal to:

$$\tau_{total} = \tau_{prox} + \tau_{ph} + \tau_{com} + \tau_{ex} \tag{9.10}$$

where τ_{total} is the total execution time, τ_{prox} is the required time for handling the proximity sensors and τ_{ph} is the necessary time for handling the sensors for pheromone detection. Moreover, τ_{com} is the required time for transmittingreceiving data for cooperation tasks and τ_{ex} is the necessary time for each robot to process all data. Each of the above parameters are strictly connected with the specifications of the available hardware modules that every robot is equipped. At this point it must be mentioned that the processor of every robot is far faster than the other sensors and thus, τ_{ex} is considered to be insignificant. Moreover, in case of a detected pheromone trail, communications between robots are not required making the creation of the paths even faster. Depending on the used sensors in either a system or a simulated system, the

method could create collision free paths at real time and regain a scattered formation after a small time period.

9.4.2 Simulator Results

For testing purposes, the proposed method was implemented in a cooperative robot team consisted of ten modified e-puck robots, using Webots. The entire team was divided into two equally numbered subgroups of robots. For this specific system, the cell length was decided to be equal to 2 cm for accuracy reasons. Moreover, every robot must cover a distance of 140 cm meaning that the configuration area is divided into a lattice with dimensions 70×70 cells. Finally, every robot moves to its final destination following the desired shortest path, that is a straight line trail, and each subgroup is forming a straight line pattern.

At the first time step, the first subgroup leaves from its initial positions, or else food sources. Each robot enables its distance sensors so that possible obstacles can be detected. Depending on the received readings, every robot decides if an obstacle is present or not. In case of a detected obstacle, the correspondent cell will be stated as an obstacle cell and by applying the appropriate CA rule, collision avoidance is achieved. If no object is present, every robot enables its ground sensors to search if any pheromone is deposited on the ground. Depending on the sensor readings, the robot decides if any of its neighboring free cells must be stated as a pheromone cell with state that represents the detected pheromone level. If a pheromone cell is present, the appropriate pheromone CA rule is applied. Obviously, no robot of the first subgroup detects any pheromone trail so cooperation tasks are applied. Every robot communicates with its neighbors and exchanges data related to its position in the grid. According to the response, each robot selects the appropriate CA rule to evolve its state to the next time step, meaning that the robot will adjust its movements. The same procedure is repeated until every robot covers the desired distance. After a period of time, the second subgroup begins its route to their final points in the configuration area following the exact same process.

Figures 9.15, 9.16 and 9.17 illustrate screenshots obtained during the evolution process. In particular, screenshots of the robot team avoiding a rectangular shaped object are shown in Fig. 9.15 while Fig. 9.16 demonstrates the avoidance of circular object. Moreover, Fig. 9.17 illustrates how robots react in a case of a dynamical environment. The presented results prove that the proposed method can produce accurate collision free paths by using simple and cheap sensors. Furthermore, the robustness and the effectiveness of the method are established.

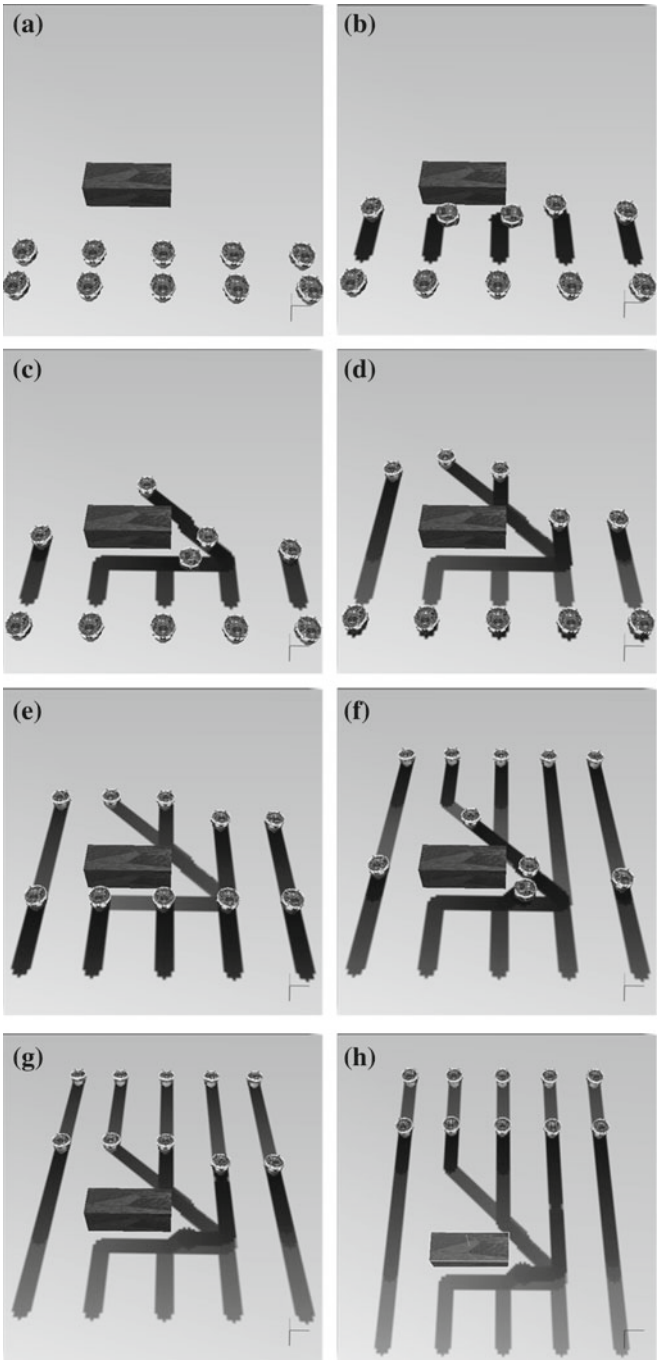


Fig. 9.15 Modified e-puck robot team avoiding a *rectangular shaped* object (from *left* to *right* and from *upper* to *lower* images), where green spots are the robots and "pheromone" trails are illustrated with lines of tones of *black*

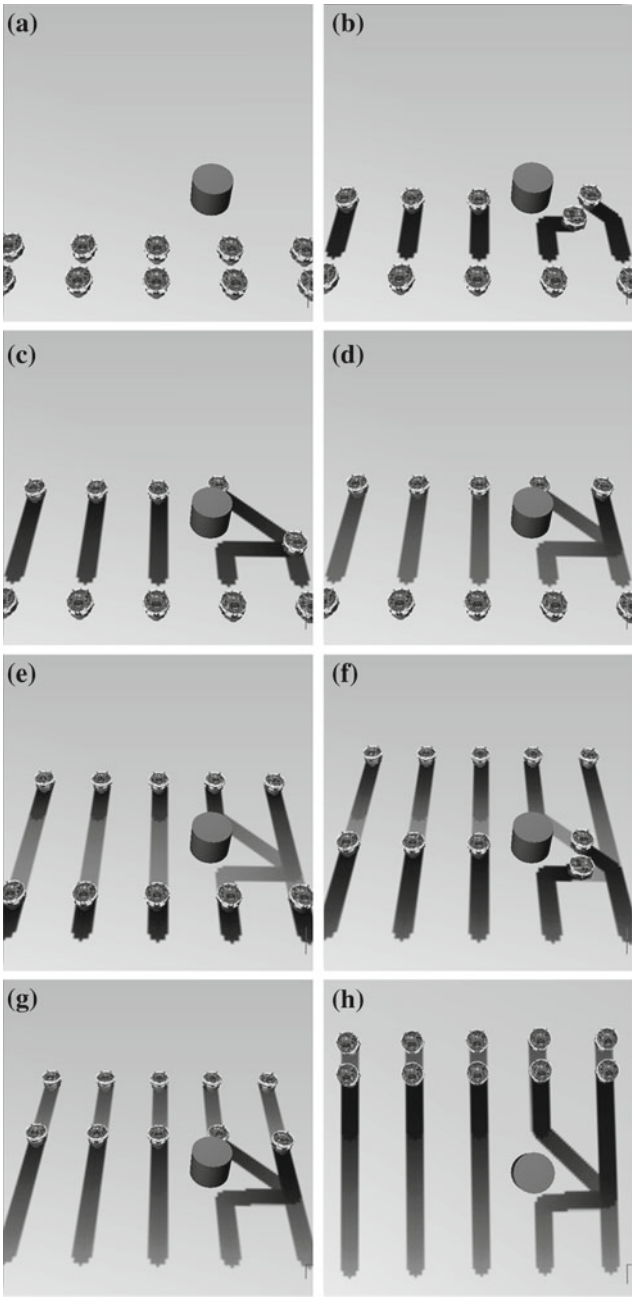


Fig. 9.16 Modified e-puck robot team avoiding a *circular shaped* object, (from *left* to *right* and from *upper* to *lower* images), where green spots are the robots and "pheromone" trails are illustrated with lines of tones of *black*

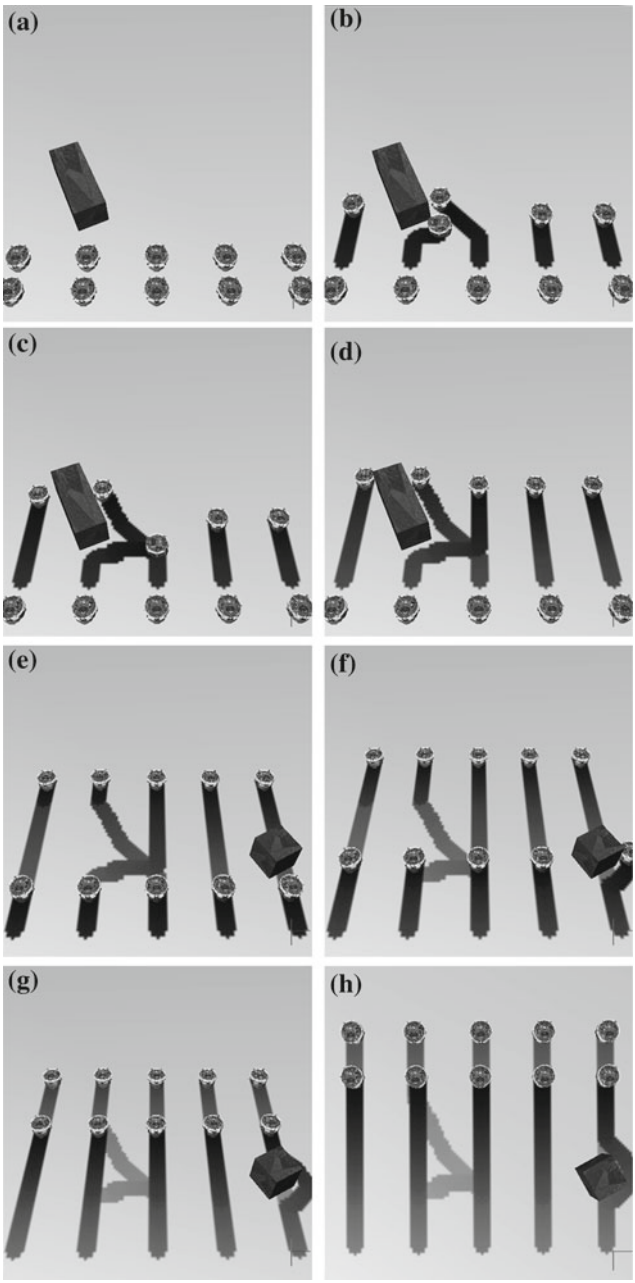


Fig. 9.17 Modified e-puck robot team avoiding dynamic moving objects, (from *left* to *right* and from *upper* to *lower* images), where *green spots* are the robots and "pheromone" trails are illustrated with lines of tones of *black*

9.5 Conclusions

In this chapter, a method for solving the path planning problem in a cooperative robot team was presented. The method is the result of a combination between CA and ACO algorithms resulting to the so called "Cellular Robotic Ants". To test the effectiveness of the method, a 2-D environment was created. The complexity of the proposed method is proportional to the CA dimensions, according to CA algorithms, and is considered to be low compared to other relative methods. Moreover, the method was implemented in a cooperative robot team using a three dimensional simulator, called Webots. For testing purposes, the under study robot team was constituted of two subgroups with five robots each. All essential sensors, that all robots must be equipped, and their direct relations with the cell length and pheromone were introduced. By some modifications on the real e-puck robot, the total amount of the required memory was reduced, thus causing the computational cost to be decreased. Therefore, the method is applicable to a real system consisting of e-puck robots but with some restrictions, as aforementioned. The accuracy of the method was tested by using two different types of objects, rectangular and circular shaped. In both cases, the method created successfully collision free paths. Presented results exhibit the effectiveness and the robustness of the method. Finally, the proposed "Cellular Robotic Ants" architecture covers the needs of self organization and autonomy of the system since no central control interferes.

References

1. Akst, J.: Send in the bots. Scientist **27**(10), 45 (2013) (Cited By since (1996))
2. Arney, T.: Dynamic path planning and execution using b-splines. In: Third International Conference on Information and Automation for Sustainability, ICIAFS 2007. pp. 1–6 (2007)
3. Barnes, L., Garcia, R., Fields, M.A., Valavanis, K.: Swarm formation control utilizing ground and aerial unmanned systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, pp. 4205–4205 (2008)
4. Beckers, R., Deneubourg, J.L., Goss, S.: Trails and u-turns in the selection of a path by the ant lasius niger. J. Theor. Biol. **159**, 397–415 (1992)
5. Belkhous, S., Azzouz, A., Saad, M., Nerguizian, C., Nerguizian, V.: A novel approach for mobile robot navigation with dynamic obstacles avoidance. J. Intell. Robotics Syst. **44**(3), 187–201 (2005)
6. Blum, C.: Ant colony optimization: Introduction and recent trends. Physics of Life Reviews **2**(4), 353–373 (2005)
7. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature **406**, 39–42 (2000)
8. Bonani, M., Raemy, X., Pugh, J., Mondana, F., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a Robot Designed for Education in Engineering. In: Proceedings of the 9th Conference on Autnomous Robot Systems and Competitions, vol. 1, pp. 59–65 (2009)
9. Brown, M., Lowe, D.: Automatic panoramic image stitching using invariant features. International Journal of Computer Vision **74**(1), 59–73 (2007)
10. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Trans. Robot. **21**(3), 376–386 (2005)

11. Charalampous, K., Amanatiadis, A., Gasteratos, A.: Efficient robot path planning in the presence of dynamically expanding obstacles. In: Sirakoulis, G., Bandini, S. (eds.) Cellular Automata. Lecture Notes in Computer Science, vol. 7495, pp. 330–339. Springer, Berlin Heidelberg (2012)
12. Charalampous, K., Kostavelis, I., Amanatiadis, A., Gasteratos, A.: Real-time robot path planning for dynamic obstacle avoidance. J. Cell. Automata Appear (2014)
13. Chen, M.J., Huang, C.H., Lee, W.L.: A fast edge-oriented algorithm for image interpolation. Image Vis. Comput. **23**(9), 791–798 (2005)
14. Chicco, G., Ionel, O.M., Porumb, R.: Electrical load pattern grouping based on centroid model with ant colony clustering. IEEE Trans. Power Syst. **28**(2), 1706–1715 (2013)
15. Conti, C., Roisenberg, M., Neto, G., Porsani, M.: Fast seismic inversion methods using ant colony optimization algorithm. IEEE Geosci. Remote Sens. Lett. **10**(5), 1119–1123 (2013)
16. Daniel, K., Nash, A., Koenig, S., Felner, A.: Theta*: any-angle path planning on grids. J. Artif. Intell. Res. **39**, 533–579 (2010)
17. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of a*. J. ACM **32**(3), 505–536 (1985)
18. Defoort, M., Floquet, T., Kokosy, A., Perruquetti, W.: Sliding-mode formation control for cooperative autonomous mobile robots. IEEE Trans. Ind. Electron. **55**(11), 3944–3953 (2008)
19. Deneubourg, J., Goss, S.: Collective patterns and decision-making. Ethol. Ecol. Evol. **1**(4), 295–311 (1989)
20. Dhiman, N.K., Deodhare, D., Khemani, D.: A review of path planning and mapping technologies for autonomous mobile robot systems. In: Proceedings of the 5th ACM COMPUTE Conference: Intelligent and Scalable System Technologies, COMPUTE '12, pp. 3:1–3:8. ACM, New York, NY, USA (2012)
21. Di Caro, G., Dorigo, M.: Antnet: Distributed stigmergetic control for communications networks. J. Artif. Int. Res. **9**(1), 317–365 (1998)
22. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Politecnico di Milano, Italy (1992)
23. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
24. Du, Z., Qu, D., Xu, F., Xu, D.: A hybrid approach for mobile robot path planning in dynamic environments. In: IEEE International Conference on Robotics and Biomimetics, ROBIO 2007, pp. 1058–1063 (2007)
25. Fax, J., Murray, R.: Information flow and cooperative control of vehicle formations. IEEE Trans. Autom. Control **49**(9), 1465–1476 (2004)
26. Fredslund, J., Mataric, M.: A general algorithm for robot formations using local sensing and minimal communication. IEEE Trans. Robot. Autom. **18**(5), 837–846 (2002)
27. Garnier, S., Combe, M., Jost, C., Theraulaz, G.: Do Ants Need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test Bed. PLoS Comput. Biol. **9**(3), e1002,903+ (2013)
28. Garnier, S., Gurcheau, A., Combe, M., Fourcassi, V., Theraulaz, G.: Path selection and foraging efficiency in argentine ant transport networks. Behav. Ecol. Sociobiol. **63**(8), 1167–1179 (2009)
29. Garnier, S., Tache, F., Combe, M., Grimal, A., Theraulaz, G.: Alice in pheromone land: An experimental setup for the study of ant-like robots. In: IEEE Swarm Intelligence Symposium, SIS 2007, pp. 37–44 (2007)
30. Ge, S.S., Fua, C.H.: Queues and artificial potential trenches for multirobot formations. IEEE Trans. Robot. **21**(4), 646–656 (2005)
31. Georgoudas, I., Sirakoulis, G., Scordilis, E., Andreadis, I.: A cellular automaton simulation tool for modelling seismicity in the region of xanthi. Environ. Model. Softw. **22**(10), 1455–1464 (2007)
32. Goss, S., Beckers, R., Deneubourg, J., Aron, S., Pasteels, J.: How trail laying and trail following can solve foraging problems for ant colonies. In: Hughes, R. (ed.) Behav.ural Mechanisms of Food Selection, NATO ASI Series, vol. 20, pp. 661–678. Springer, Berlin (1990)

33. Grassé, P.P.: La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes Soc. **6**(1), 41–80 (1959)
34. Herianto Kurabayashi, D.: Realization of an artificial pheromone system in random data carriers using rfid tags for autonomous navigation. In: IEEE International Conference on Robotics and Automation, ICRA '09, pp. 2288–2293 (2009)
35. Herianto Sakakibara: T., Kurabayashi, D.: Artificial pheromone system using RFID for navigation of autonomous robots. J. Bionic Eng. **4**(4), 245–253 (2007)
36. Huang, W.H., Fajen, B.R., Fink, J.R., Warren, W.H.: Visual navigation and obstacle avoidance using a steering potential function. Robot. Auton. Syst. **54**, 288–299 (2006)
37. Ioannidis, K., Sirakoulis, G.C., Andreadis, I.: Cellular automata-based architecture for cooperative miniature robots. J Cell. Automata **8**(1–2), 91–111 (2013)
38. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall Inc, Upper Saddle River (1989)
39. Keys, R.: Cubic convolution interpolation for digital image processing. IEEE Trans. Acoust. Speech Signal Process. **29**(6), 1153–1160 (1981)
40. Konstantinidis K., Andreadis I., Sirakoulis G.C.: Chapter 3—application of artificial intelligence methods to content-based image retrieval. In: P.W. Hawkes (ed.) Advances in Imaging and Electron Physics, Advances in Imaging and Electron Physics, vol. 169, pp. 99–145. Elsevier, Amsterdam (2011)
41. Konstantinidis, K., Sirakoulis, G., Andreadis, I.: Design and implementation of a fuzzy-modified ant colony hardware structure for image retrieval. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **39**(5), 520–533 (2009)
42. Latombe, J.C.: Robot Motion Plann. Kluwer Academic Publishers, Norwell (1991)
43. Lee, T.L., Wu, C.J.: Fuzzy motion planning of mobile robots in unknown environments. J. Intell. Robotics Syst. **37**(2), 177–191 (2003)
44. Li, X., Orchard, M.: New edge directed interpolation. In: Proceedings of the International Conference on Image Processing, vol. 2, pp. 311–314 (2000)
45. Lin, C.T., Fan, K.W., Pu, H.C., Lu, S.M., Liang, S.F.: An hvs-directed neural-network-based image resolution enhancement scheme for image resizing. IEEE Trans. Fuzzy Syst. **15**(4), 605–615 (2007)
46. Liu, J., Wu, J.: Multi-Agent Robotic Systems. CRC Press, Boca Raton (2001)
47. Marchese, F.: Multiple mobile robots path-planning with MCA. In: International Conference on Autonomic and Autonomous Systems, ICAS '06, pp. 56–56 (2006)
48. Marchese, F.M.: A directional diffusion algorithm on cellular automata for robot path-planning. Future Gener. Comput. Syst. 18(7), 983–994 (2002). Selected papers from CA2000 (6th International Workshop on Cellular Automata of IFIP working group 1.5, Osaka, Japan, 21–22 Aug 2000) and ACRI2000 (4th International Conference on Cellular Automata in Research and Industry, Karlsruhe, Germany, 4–6 Oct
49. Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Trans. Evol. Comput. **11**(5), 651–665 (2007)
50. Mastellone, S., Stipanovic, D, Spong, M.: Remote formation control and collision avoidance for multi-agent nonholonomic systems. In: IEEE International Conference on Robotics and Automation, pp. 1062–1067 (2007)
51. Michel, O.: Webots: Professional mobile robot simulation. J. Adv. Robot. Syst. **1**(1), 39–42 (2004)
52. Muresan, D., Parks, T.: Adaptively quadratic (aqua) image interpolation. IEEE Trans. Image Process. **13**(5), 690–698 (2004)
53. Murphy, R.: Human-robot interaction in rescue robotics. IEEE Trans. Syst. Man Cyber. Part C Appl. Rev. **34**(2), 138–153 (2004)
54. Omohundro, S.: Modelling cellular automata with partial differential equations. Physica D: Nonlinear Phenomena **10**(1–2), 128–134 (1984)
55. Patnaik, S., Karibasappa, K.: Motion planning of an intelligent robot using ga motivated temporal associative memory. Appl. Artif. Intell. **19**(5), 515–534 (2005)

56. Progias, P., Sirakoulis, G.C.: An fpga processor for modelling wildfire spreading. Math. Comput. Modell. **57**(5-6), 1436–1452 (2013)
57. Recio, G., Martin, E., Estebanez, C., Saez, Y.: Antbot: Ant colonies for video games. IEEE Trans. Comput. Intell. AI Game. **4**(4), 295–308 (2012)
58. Russell, R.A.: Heat trails as short-lived navigational markers for mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 3534–3539 (1997)
59. Shen, H., Zhang, L., Huang, B., Li, P.: A map approach for joint motion estimation, segmentation, and super resolution. IEEE Trans. Image Process. **16**(2), 479–490 (2007)
60. Stentz, A.: The focussed d* algorithm for real-time replanning. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95, vol. 2, pp. 1652–1659. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
61. Tan, K.C., Tan, K., Lee, T., Zhao, S., Chen, Y.J.: Autonomous robot navigation based on fuzzy sensor fusion and reinforcement learning. In: Proceedings of the IEEE International Symposium on Intelligent Control, pp. 182–187 (2002)
62. Toffoli, T.: Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. Physica D: Nonlinear Phenomena **10**(1–2), 117–127 (1984)
63. Tzionas, P., Thanailakis, A., Tsalides, P.: Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata. IEEE Trans. Robot. Autom. **13**(2), 237–250 (1997)
64. Ulam, S.: Random processes and transformations. Int. Congr. Math. **2**, 264–275 (1952)
65. Von Neumann, J., Burks, A. et al.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
66. Wang, C., Soh, Y., Wang, H., Wang, H.: A hierarchical genetic algorithm for path planning in a static environment with obstacles. In: IEEE CCECE2002 Canadian Conference on Electrical and Computer Engineering, vol. 3, pp. 1652–1657 (2002)
67. Willms, A., Yang, S.X.: An efficient dynamic system for real-time robot-path planning. IEEE Trans. Syst. Man Cybern. Part B Cybern. **36**(4), 755–766 (2006)
68. Willms, A.R., Yang, S.X.: An efficient dynamic system for real-time robot-path planning. IEEE Trans. Syst. Man Cybern. Part B **36**(4), 755–766 (2006)
69. Yang, S.X., Luo, C.: A neural network approach to complete coverage path planning. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**(1), 718–724 (2004)
70. Zheng, T., Zhao, X.: Research on optimized multiple robots path planning and task allocation approach. In: IEEE International Conference on Robotics and Biomimetics, ROBIO '06, pp. 1408–1413 (2006)
71. Zhong, Y., Shirinzadeh, B., Tian, Y.: A new neural network for robot path planning. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1361–1366 (2008)

Chapter 10
Employing Cellular Automata for Shaping Accurate Morphology Maps Using Scattered Data from Robotics' Missions

Athanasios Ch. Kapoutsis, Savvas A. Chatzichristofis, Georgios Ch. Sirakoulis, Lefteris Doitsidis and Elias B. Kosmatopoulos

Abstract Accurate maps are essential in the case of robot teams, so that they can operate autonomously and accomplish their tasks efficiently. In this work we present an approach which allows the generation of detailed maps, suitable for robot navigation, from a mesh of sparse points using Cellular Automata and simple evolutions rules. The entire map area can be considered as a 2D Cellular Automaton (CA) where the value at each CA cell represents the height of the ground in the corresponding coordinates. The set of measurements form the original state of the CA. The CA rules are responsible for generating the intermediate heights among the real measurements. The proposed method can automatically adjust its rules, so as to encapture local morphological attributes, using a pre-processing procedure in the set of measurements. The main advantage of the proposed approach is the ability to maintain an accurately reconstruction even in cases where the number of measurements are significant reduced. Experiments have been conducted employing data collected from two totally different real-word environments. In the first case the proposed approach is applied, so as to build a detailed map of a large unknown underwater area in Oporto, Portugal. The second case concerns data collected by a team of aerial robots in real experiments in an area near Zurich, Switzerland and is also used for the evaluation of the approach. The data collected, in the two aforementioned cases, are extracted using different kind of sensors and robots, thus demonstrating the applicability of

The research leading to these results has received funding from the European Communities Seventh Framework Programme (FP7/2007–2013) under grant agreements n. 270180 (NOPTILUS)

A.Ch. Kapoutsis (✉) · S.A. Chatzichristofis · G.Ch. Sirakoulis · E.B. Kosmatopoulos
Department of Electrical and Computer Engineering, Democritus University of Thrace,
67100 Xanthi, GR, Greece
e-mail: akapouts@ee.duth.gr

A.Ch. Kapoutsis · S.A. Chatzichristofis · G.Ch. Sirakoulis · L. Doitsidis · E.B. Kosmatopoulos
Telematics Institute, Center for Research and Technology, Hellas (ITI-CERTH),
57001 Thessaloniki, Greece

L. Doitsidis
Department of Electronic Engineering, Technological Educational Institute of Crete,
73100 Chania, GR, Greece

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_10

our approach in different kind of devices. The proposed method outperforms the performance of other well-known methods in literature thus enabling its application for real robot navigation.

10.1 Introduction

Scattered data interpolation refers to the problem of generating the intermediate values through a non-uniform, unpredictable distribution of data samples. This numerical analysis method can be adapted in a variety of engineering fields where data is often measured or produced at random and irregular positions. The goal of interpolation is to find the best way to propagate the data, finding an underlying function [1] or utilizing the information of the neighborhood [2] and etc., onto all positions in the domain.

There are three principal sources of scattered data: measured values of physical quantities, experimental results and computational values [3]. This chapter focuses in the investigation of the first category without losing the ability of direct adjustment in other types of applications. Non-uniform measured values of physical quantities are collected in geology, meteorology, oceanography, cartography, mining, etc. Although our method applies in any of the previous categories we limit our presentation to measurement data obtained by robot teams.

In general, a key element to the successful operation of a robot team is the ability to perceive the environment in which it operates and therefore be able to function with the highest level of autonomy. Currently several types of robots including ground [4], aerial [5], surface or underwater robots [6] or even heterogeneous teams consisted of different type of robots [7] are deployed in different type of missions utilizing a diverse set of sensors. In all cases the key questions is *how the data gathered by the team members, will be processed and transformed into meaningful information, in the form of maps* so that they can be used by the robots. Usually the data collected are in the form of scattered and often noisy data.

In recent literature numerous applications of robots used for mapping of regions of interest (see e.g., Fig. 10.1) are reported. In the case of Micro Aerial Vehicles (MAVs), they have been used both in indoors [8] or outdoors [9, 10] environments using laser rangefinder sensor and a front-looking stereo camera as the main sensor respectively. A fully autonomous system using a team of MAVs has been used to construct maps of an unknown environment using a state-of-the-art visual-SLAM algorithm which tracks the pose of the camera while simultaneously and autonomously, building an incremental map of the surrounding environment [11].

In the case of underwater missions the state-of-the-art sensors, for Autonomous Underwater Vehicles and for mapping the sea-bottom, are bathymeters (sonars) or range scans [6, 12]. Furthermore, one of the sensors that is in common to all Autonomous Ground Vehicles is the sensor to perceive the environment and their movement (range sensing devices) [4].

Fig. 10.1 Real life applications of teams of Autonomous Vehicles, operating under different environment, constructed under different design architectures. **a** Unmanned Aerial Vehicles (*UAVs*) in continuous infrastructure monitoring to prevent accidents [5]. **b** Autonomous Underwater Vehicles (*AUVs*) for underwater archeology and post-disaster infrastructure inspection. **c** Autonomous Ground Vehicles (*AGVs*) during ground surveillance task

In all the aforementioned cases the vehicles' sensors produce either directly or after some processing a pool of scattered measurements of the environment in which they are operating. The elaboration of these measurements can be done either on-line or off-line. A successful demonstration of the on-line case is presented in [13] were a team of Autonomous Underwater Vehicles (AUVs) has efficiently and fully-autonomously navigated in a dynamic environment. In the off-line scenario, the robots have to follow a predefined trajectory gathering the corresponding points. After the completion of the mission the interpolation methods are used to produce the desired map. Despite the fact that, in the off-line scenario, a a-priori information

about the exact location of the measurements could be used in order to improved the performance of the interpolation process, crucial information, about the specific morphology of the area, may be lost.

There is, no unique solution to the interpolation problem, resulting in different fields when alternative techniques are applied to the same discrete data set. Since the interpolation methods in bibliography are numerous, we have used as evaluation methods four of the most common and most general applicable ones:

1. **Linear**
   Linear interpolation is the simplest method of getting values at positions between the data points. The points are simply joined by straight line segments. Each segment (bounded by two data points) can be interpolated independently. The parameter *mu* defines where to estimate the value on the interpolated line, it is 0 at the first point and 1 and the second point. For interpolated values between the two points *mu* ranges between 0 and 1.

$$y(x) = y_1 \times (1 - mu) + y_2 \times mu \tag{10.1}$$

2. **Nearest Neighbors**
   The nearest neighbors (NN) method predicts the value of an attribute at an arbitrary point based on the value of the nearest sample by drawing perpendicular bisectors between sampled points (n), forming such as Thiessen (or Dirichlet/Voronoi) polygons ($V_i, i = 1, 2, \ldots, n$). This produces one polygon per sample and the sample is located in the center of the polygon, such that in each polygon all points are nearer to its enclosed sample point than to any other sample points [14–16]. The estimations of the attribute at arbitrary points within polygon V_i are the measured value at the nearest single sampled data point x_i that is $\hat{z}(x_0) = z(x_i)$. The weights are:

$$\lambda_i = \begin{cases} 1 & \text{if } x_i \in V_i \\ 0 & \text{otherwise} \end{cases} \tag{10.2}$$

   All points (or locations) within each polygon are assigned the same value [15, 16]. A number of algorithms exist to generate the polygons [17], including pycnophylactic interpolation [18].
3. **Natural**
   The natural neighbors method was introduced by Sibson (1981). It combines the best features of NN and Triangular Irregular Network [16]. The first step is a triangulation of the data by Delauneys method, in which the apices of the triangles are the sample points in adjacent Thiessen polygons. This triangulation is unique except where the data are on a regular rectangular grid. To estimate the value of a point, it is inserted into the tessellation and then its value is determined by sample points within its bounding polygons. For each neighbors, the area of the portion of its original polygon that became incorporated in the tile of the

new point is calculated. These areas are scaled to sum 1 and are used as weights for the corresponding samples [16]. This method can provide a more smooth approximation to the underlying "true" function.

4. **Cubic**
   A cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set of N control points. The polynomials describe pieces of a line or surface (i.e., they are fitted to a small number of data points exactly) and are fitted together so that they join smoothly [16, 18]. The places where the pieces join are called knots. The choice of knots is arbitrary and may have a dramatic impact on the estimation [18]. Splines with few knots are generally smoother than splines with many knots; however, increasing the number of knots usually increases the fit of the spline function to the data. Knots give the curve freedom to bend to more closely follow the data.

In this work we propose a Cellular Automata (CA) based method for shaping accurate morphology maps using scattered data collected from multi robot teams. CA have attracted researchers from several disciplines (e.g., from the field of robotics [19, 20], image processing [21, 22] and environmental modelling [23]) and a large number of scientific papers are published every year.

CA, initially were proposed as models of physical systems, where space and time are discrete and interactions are local, by von Neumann [24]. Any physical system satisfying differential equations may be approximated by a CA, by introducing finite differences and discrete variables [25–31]. Additionally, CA are one of the computational structures best suited for a VLSI realization [32–35]. The CA architecture offers a number of advantages and beneficial features such as simplicity, regularity, ease of mask generation, silicon-area utilization, and locality of interconnections [26, 33].

In order to evaluate the proposed approach, experiments conducted employing real-word data collected from two different types of robot teams. Initially, using the proposed CA based method for shaping accurate morphology maps, a detailed map of a large unknown underwater area in Oporto, Portugal was constructed. In the sequel, the proposed approach generates a detailed map using data collected from an area near Zurich, Switzerland, by a team of aerial robots. It is worth noting that the collected data, in the two experimental setups, are captured employing different type of sensors. In both cases, the proposed CA based method outperforms the performance of several other well-known methods from the literature.

The rest of the chapter is organized as follows. Section 10.2.1 demonstrates the problem of scattered data interpolation in strict notation. In Sect. 10.2.2, we demonstrate the exact steps of the proposed methodology using CA. Section 10.3 presents a series of experiments carried out with measurements from real robot systems. Conclusions and future steps are given in Sect. 10.4.

10.2 CA Based Methodology for Shaping Morphology Maps Using Scattered Data

10.2.1 Problem Formulation

Without loss of generality, we can assume that the area to be mapped is constrained within a rectangle in the (x, y)-coordinates, i.e., the mobile robots are called to map the area constrained in the (x, y)-coordinates as follows:

$$\mathcal{U} = \left\{ x, y : x \in [x_{min}, x_{max}], y \in [y_{min}, y_{max}] \right\} \tag{10.3}$$

This rectangle can be divided into discrete cells, in such a way that if all the values in each cell are known, the representation of this rectangle would approximate the real surface.

$$\begin{aligned} x_{i+1} &= x_i + \Delta x, \Delta x = \tfrac{(x_{max} - x_{min})}{L} \\ y_{i+1} &= y_i + \Delta y, \Delta y = \tfrac{(y_{max} - y_{min})}{M} \end{aligned} \tag{10.4}$$

where L and M denote the desirable discretization in x and y axis correspondingly. The goal is from an, arbitrarily located, set of data $(x_i, y_i, f_i), i = 1, \ldots, N$ that represent the error-free[1] measurements taken from different type of sensors, to generate the values $(x, y) \in \mathcal{U}$ in all the cells of the rectangle.

10.2.2 Proposed Methodology

The basic steps of the proposed CA methodology are given as follows:

STEP 1: The map area is divided into a matrix, and for now on it will be referred as $\mathcal{C}$, of identical square cells that represented by a CA, where each cell of the map is considered as a CA cell.

STEP 2: In the second step is applied the registration between the measurements' data and the corresponding CA cells. In other words every set (x_i, y_i, f_i) of measurements has to be placed in the appropriate cell $\mathcal{C}(x_i, y_i) = f_i$. After this step we have defined the dimensions of our CA and its initial conditions.

STEP 3: The evolution rules of a m CA cell, where $m : (m_x, m_y)$, are chosen as a combination of two different approaches. On the one hand, a direct "propagation" of the information is applied, following the Moore's neighbor (Eq. 10.5), around the initially known cells. Please note that the logical expression: $||i - m_x|| = 1$ or

[1] We will assume that the robot's measurements are filtered and free of bias/Gaussian noise. It has to be emphasized that the proposed approach can be extended to deal with noisy data giving weights about the confidence level of the measurement's accuracy.

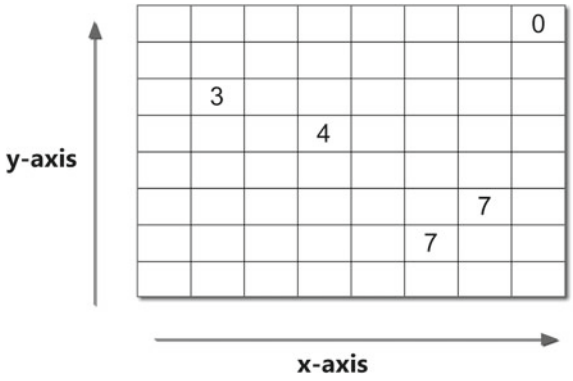


Fig. 10.2 The initial conditions of CA

$||j - m_y|| = 1$ has to be true in order to follow the Moore's neighborhood. If a cell is affected by more than one value in the same step, a simple average is applied (see Fig. 10.3).

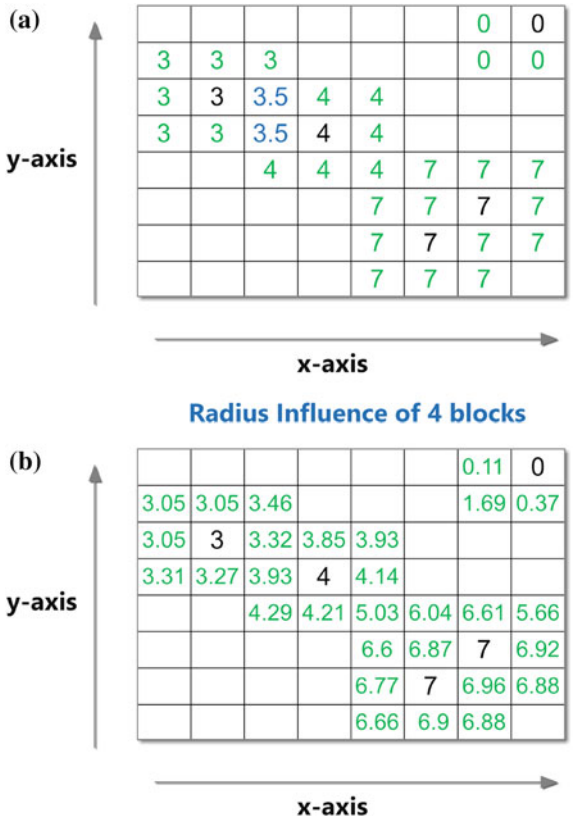


Fig. 10.3 The evolution rules. **a** Direct "propagation". **b** Using Remote Information

$$\mathcal{C}_1^{t_i}(m_x, m_y) = \begin{cases} \mathcal{C}^{t_i}(i, j) & \text{if } \mathcal{C}_1^{t_i-1}(m_x, m_y) = 0 \\ \mathcal{C}_1^{t_i-1}(m_x, m_y) & \text{otherwise} \end{cases} \tag{10.5}$$

On the other hand, an evolution rule using information of the closest neighbors is applied. We calculate the estimated value using a weighted average of the neighbors that are in a pre-specified (see 10.2.3) "radius of influence" (*RD*) as shown in Eq. 10.6:

$$\mathcal{C}_2^{t_i}(m_x, m_y) = \frac{\sum_{k=1}^{n} \mathcal{C}^{t_{i-1}}\left(k_x, k_y\right) / M_d(k, m)^2}{\sum_{k=1}^{n} 1/M_d(k, m)^2} \tag{10.6}$$

STEP 4: Subsequently, a merging procedure is applied in order to render the final value of the estimated cell, which, can be represented as follows:

$$\mathcal{C}^{t_i}(m_x, m_y) = a \times \mathcal{C}_1^{t_i}(m_x, m_y) + (1 - a) \times \mathcal{C}_2^{t_i}(m_x, m_y) \tag{10.7}$$

Here, a servers as *smoothing factor* to give more/less weight to the one term against the other.

STEP 5: Repeat *STEP 3–4* until every cell of the CA obtain an estimation about its height.

10.2.3 Define Adaptively the "Radius of Influence"

The *RD* corresponds to the maximum distance that is allowed between the current CA cell and every other cell with value already calculated. It defines how far "travels" the information, from the measurements, on the terrain. There is no global value for the *RD* that can be applied in every map, e.g., if the area to be mapped was "flat", probably a good strategy would be to choose a "big" value for the *RD*. Information about the morphology of the area, and thus about the *RD*, can be derived by exploiting the distribution of the measurements' set. The following steps are describing the dynamic adjustment of the *RD*, utilizing the robot's measurements.

STEP 1: The measurements sets have specific attributes, that could differ from a sub-area to another. Initially these measurements data have to be classified, in an optimal manner. To keep the analysis as general as possible, it will be considered that the number of classes is unknown, and have to be investigated as the initial measurements change. Based on the above we can formulate the following optimization problem,

$$\begin{aligned} &\underset{k}{\text{minimize}}\ \mathcal{F}(k) = \sum_{i=1}^{N} || \left[x_c\ y_c\ h_c\right]^i - \left[x_i\ y_i\ h_i\right] || \\ &\text{subject to } k \geq 2 \end{aligned} \tag{10.8}$$

where N is the number of the measurements, k denotes the number of centroids, $[x_i\ y_i\ h_i]$ denotes the i-th measurement vector and $[x_c\ y_c\ h_c]^i$ the centroid of the

class, where with the current centroids' selection, belongs the i-th measurement. The cost function $\mathcal{F}(k)$ can be separated in two terms.

Whenever an updated value, about the number of classes, is calculated, the algorithm K-means is called to find the 3 dimension vector of each centroid. Only when the iterative procedure of K-means ends, the cost function $\mathcal{F}(k)$ (Eq. 10.8) is calculated again to evaluate the difference in classification with the modified number of centroids.

Taking into account that the dimensions of the centroids are in three dimensional space, the above procedure can be completed in a reasonable time using an optimization algorithm such as Hill climbing.

STEP 2: Having defined the optimal number of classes k_{opt} and the corresponding centroids (using K-means) we can now proceed to the final calculation about the *RD*.

$$RD = \frac{\mathcal{F}(k_{opt})}{N} \tag{10.9}$$

The *RD* corresponds to the influence of the current cell in its neighborhood. If we calculate the average of distances between all the measurements and their centroids (Eq. 10.9), we can have a rough estimation about the spatial influence of every cell around its neighborhood.

10.3 Experiments

In this section we validate the proposed approach using real data collected by two different types of robotic devices using different sensors. In both cases the robots were collecting the data in order to construct a map to assist them in performing a predefined mission. The first case refers to data collected from Oporto's harbor area using bathymetry sensors [36], while the second test case refers to data collected using a camera mounted on a single aerial robot [11]. The diversity of the data collected from two different type of devices is used as a proof of concept of the generality and applicability of our approach.

10.3.1 Underwater Scenario—Oporto harbor

Using the method described in detail in Sect. 10.2.2, we have reconstructed the undersea morphology of a sub-region of Oporto's harbor. The underwater region covers an area of 200 × 200 points spaced by 5 m. To acquire full knowledge of the seafloor we should have known the value in each of the 40,000 points arising from the previous grid. In Fig. 10.4 we present the real morphology of the area and the information which will be used as ground truth to evaluate our approach. In a realistic scenario the robot team, in this case a team of autonomous underwater vehicles, would have returned after the completion of a mission with a data set much smaller

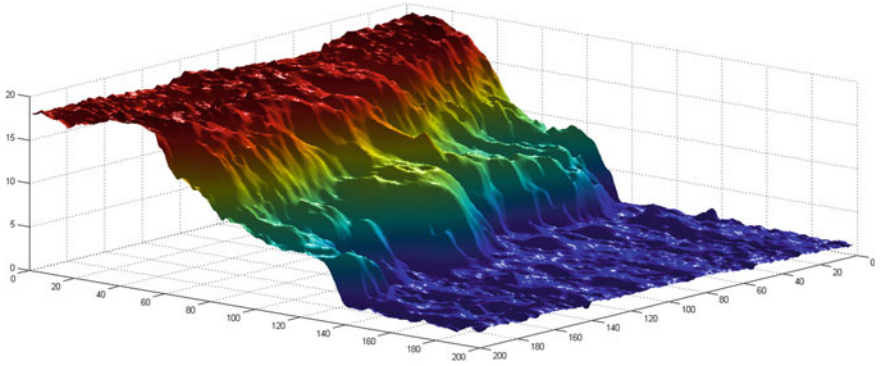


Fig. 10.4 Ground truth—40,000 points—real representation of the operation area

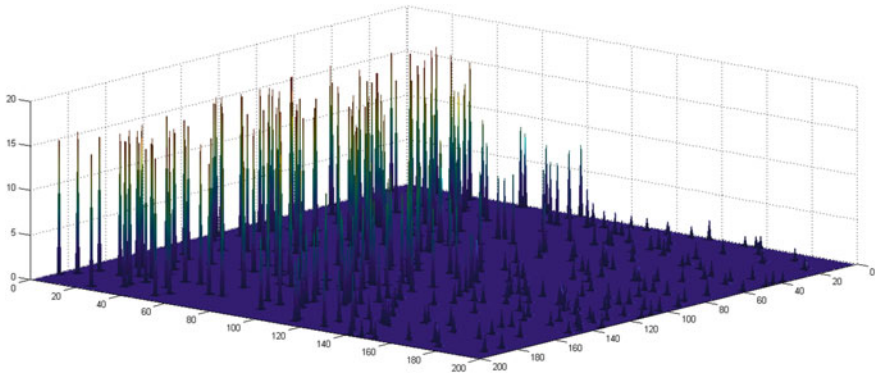


Fig. 10.5 Initial measurements—308 points gathered by swarm of robots

than the one needed to have a detailed representation of the environment monitored. Figure 10.5 depicts the visualization of real scenario where the AUVs gathered 308 measurements points. It is worth-noticing that this subset (derived from robot's measurements) constitutes only the 0.77 % [2] of the total area to be mapped, and for that is considered a severe experiment.

The details of the CA environment are as follows:

- 2-D CA with 200×200 cells
- The initial condition of the CA are the 308 measurements as depicted in Fig. 10.5 and the unknown cells are filled with -1
- The evolution rules are as explained in the previous section (Eq. 10.7)

[2] Note that now and in the next experiments there has not been any analysis about the distribution that is followed by the measurements. Different modalities, like different number of robots or different type of sensors, etc., will lead to a different data distribution. The above problem is tackled by conducted the same experiment 500 times with the initial measurements stochastically changed and keep the average of error.

The Fig. 10.6 demonstrates the incremental evolution of the CA over the time. The whole procedure, even in this case with the extremely sparse measurements data, is completed in 20 time-steps, renders the procedure directly applicable to real-time interpolation systems.

As evaluation, the Fig. 10.7 illustrates the results of different interpolation methods for the same data sets. The visual superiority of the proposed method is appeared also in extended simulations, where a batch of 500 iterations, with different initial measurements (conditions), per method is conducted and the average of L^2 error with ground-truth is calculated. The results are shown in Table 10.1.

10.3.2 Aerial robots Scenario

To test the efficiency and stability of the proposed approach, regardless of the area which is called to reconstruct the morphology, we have also tested it in data gathered from aerial robots. In this case we will demonstrate the reconstruction of a "village like" area based on data collected in an area near Zurich, Switzerland. The initial data and the respective map were collected using a state-of-the-art visual-SLAM algorithm which tracks the pose of the camera while simultaneously and autonomously, building an incremental map of the surrounding environment. More details regarding the extraction methodology are given in [8, 37].

This area is consisted of ruins and small urban structures and for that has high rate of inhomogeneity and the transitions among different sub-areas are often steep. For those reasons the interpolation in measurements' data consists in a very challenging task. In Fig. 10.8 is presented the real surface with 13,855 points Fig. 10.8a and the reconstructed one from 462 (3.3 %) measurements: using the proposed method Fig. 10.8b, while in Fig. 10.9 as evaluation we illustrate the results of the different interpolation methods for the same data sets. The average of L^2-*Norm* for this initial configuration for each method is presented in the second row of Table 10.2. The Table 10.2 is an analysis of the impact that has the reduction of the initial measurements in the final result of the interpolation procedure.

As robots are reducing their sampling rates of their sensors or operating for less time the pool of gathered measurements will be smaller and thus the interpolation procedure will have less accuracy, regardless of the choice of the interpolation method and that's something that is imprinted in the Table 10.2. The proposed method achieves better performance not only as weighted average, but in every different configuration in number of measurements. This property can be achieved because the proposed method can adapt its parameters in the available data set (Sect. 10.2.3) and finally is able to manage to elaborate them in a efficient fashion.

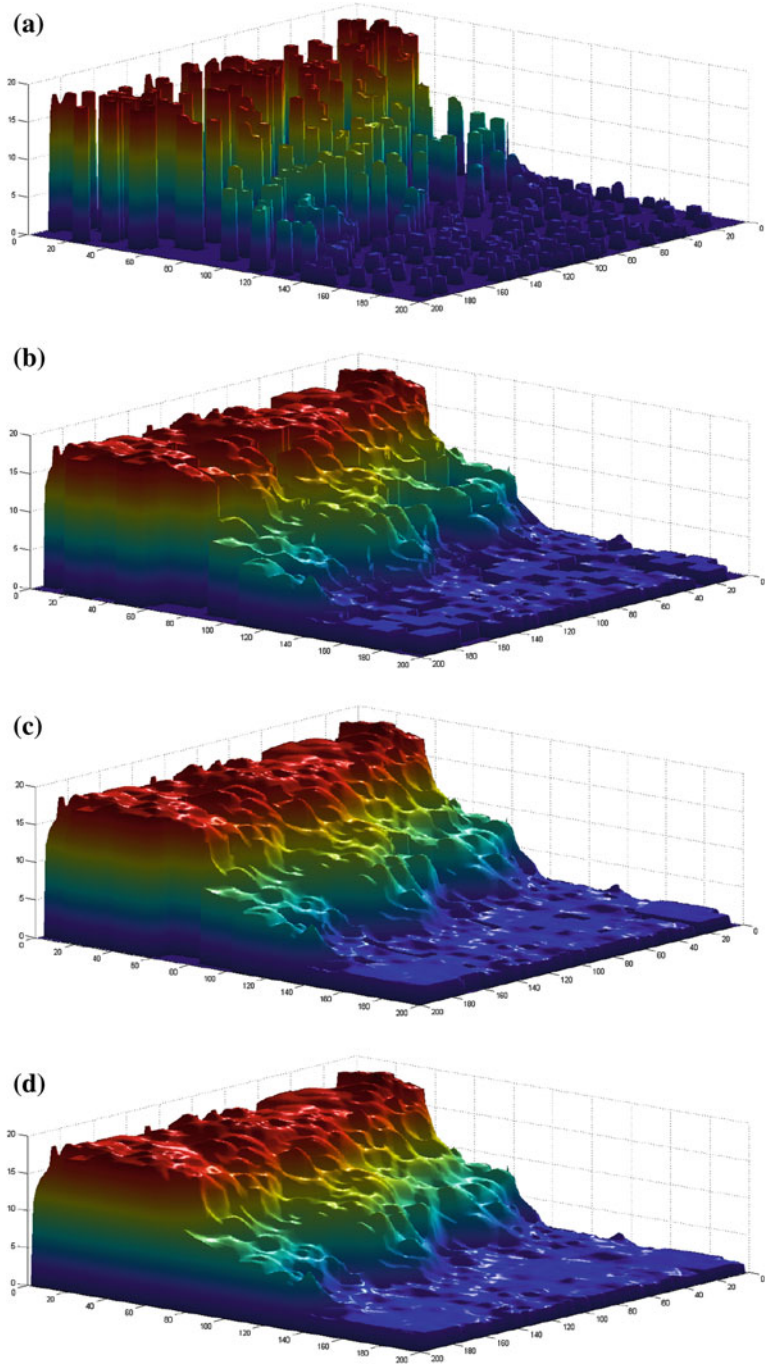


Fig. 10.6 Incremental evolution based on the CA methodology. The figures illustrate the progress at 25, 50, 75 and 100 %, correspondingly, of the CA process

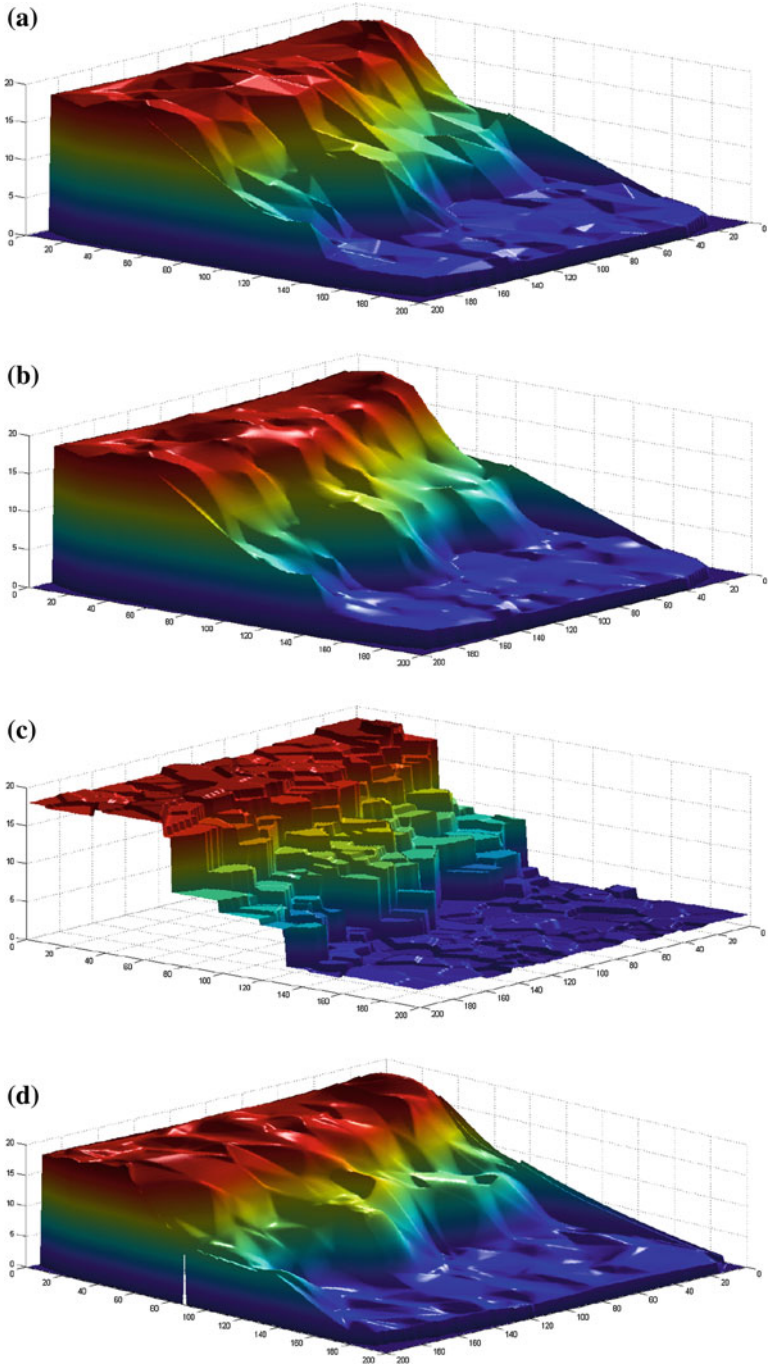


Fig. 10.7 Results using other interpolation methods. **a** Linear, **b** Natural, **c** Nearest neighbors, **d** Cubic

Table 10.1 *L^2-Norm* between the ground truth and the constructed map for each method for each number of initial measurements

L^2-Norm	Linear	Natural	Nearest neighbors	Cubic	Proposed
Weighted average	340.1553	340.8651	288.520	338.520	266.9886

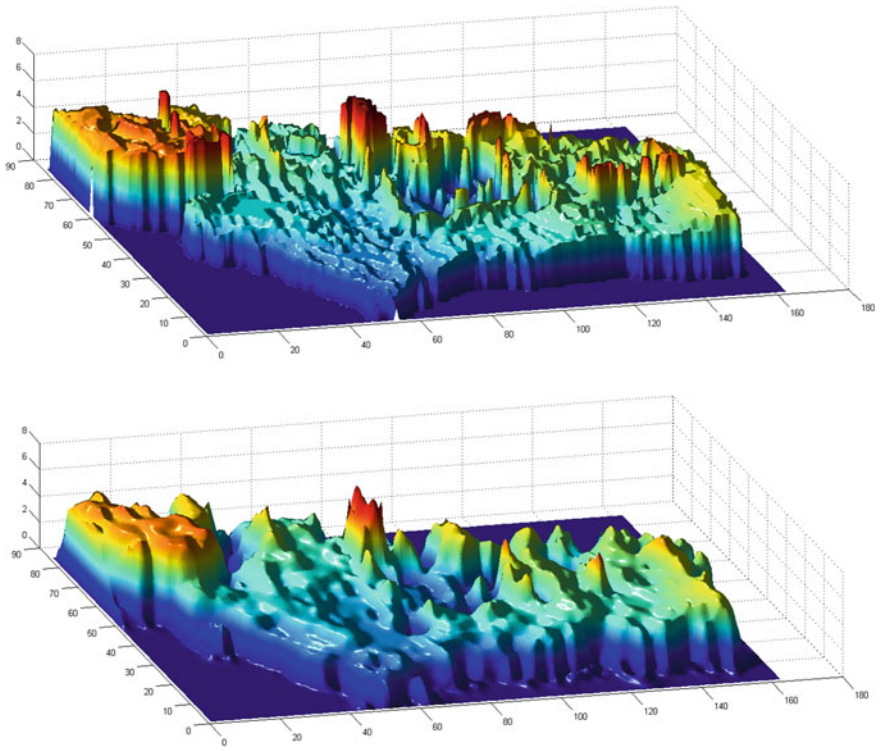


Fig. 10.8 Zurich area—interpolation results over 462 points (3.3 %), **a** Ground truth **b** Proposed methodology using CA

10.4 Conclusions and Future Work

In this chapter, a novel method is presented using CA for scattered data interpolation. We have successfully demonstrated the composition of morphology maps from sensor's measurements that outperform the most common used ones in all the test-cases. The efficiency of the methodology relies, both on the ability of CA to efficiently process elements that are arranged in a regular grid of identical cells, and on the adaptability on the local morphology of each region, analyzing the variety in measurements.

We are interested in considering situations where the robots' measurements contain errors. In this case, the problem becomes even harder since the CA now have

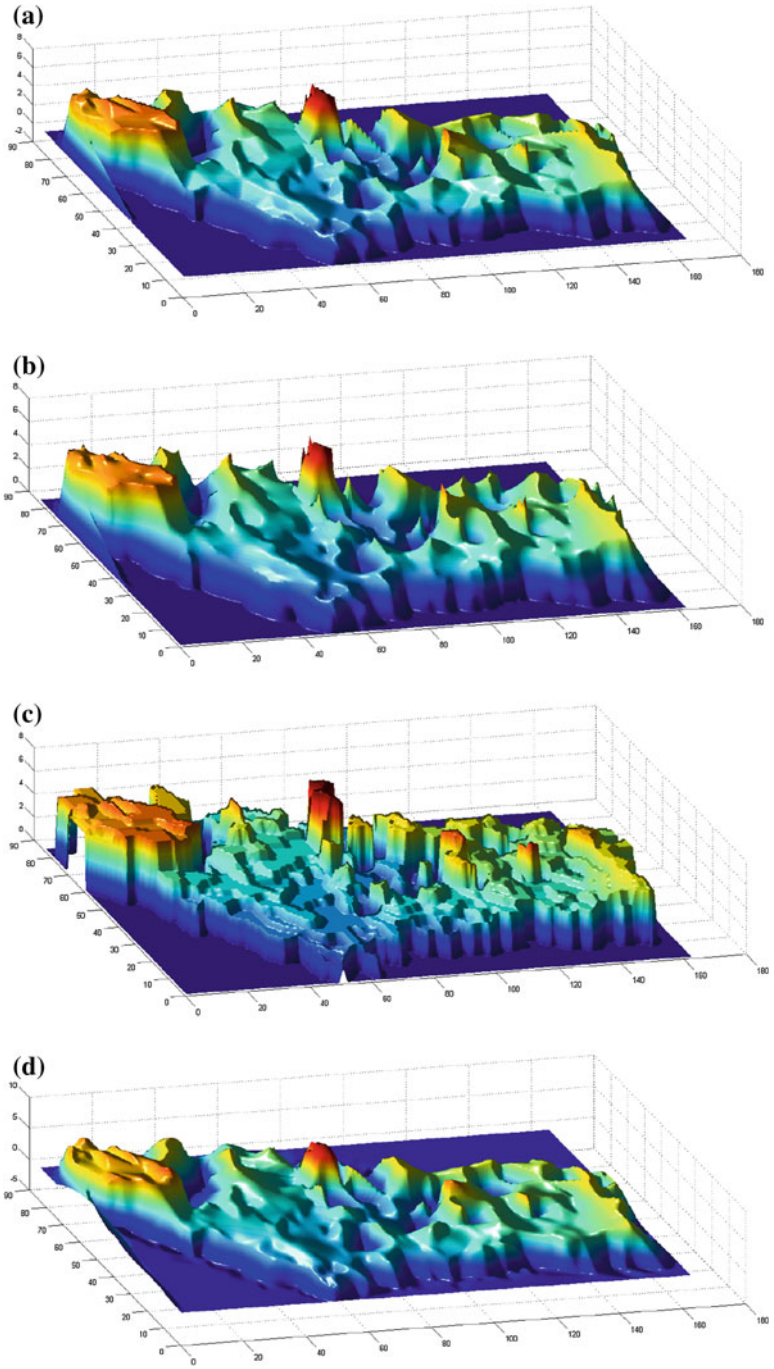


Fig. 10.9 Zurich area—Results using other interpolation methods, namely **a** Linear, **b** Natural, **c** Nearest neighbors, **d** Cubic

Table 10.2 *L^2-Norm* between the ground truth and the constructed map for each method for each number of initial measurements

Initial points %	Linear	Natural	Nearest neighbors	Cubic	Proposed
1386 (10)	361.117	343.837	407.426	355.361	338.004
462 (3.3)	499.623	477.9376	562.482	507.109	463.655
277 (2)	593.827	571.870	656.690	602.876	544.507
139 (1)	754.819	734.043	826.645	772.014	699.616
69 (0.5)	959.421	938.274	1032.046	933.780	866.569
46 (0.33)	1108.377	1086.865	1140.229	1079.947	928.137
Weighted average	*720.022*	*699.247*	*778.214*	*715.578*	*657.203*

to solve two problems. In the case CA will have to face a dual problem, since they will have to identify which measurements are useful in the process and which have to be ignored or corrected and also must be able to make an estimation, followed the proposed methodology, about the morphology of the terrain, in real-time to keep its directly applicable nature. We would like also to investigate scenarios in which the objective is to build the morphology map of sub-region, where the robots do not visit at all (extrapolation) or in which the environment changes over time.

References

1. Gasca, M., Sauer, T.: Polynomial interpolation in several variables. Adv. Comput. Math. **12**(4), 377–410 (2000)
2. Lehmann, T.M., Gonner, C., Spitzer, K.: Survey: Interpolation methods in medical image processing. Med. Imaging IEEE Trans. **18**(11), 1049–1075 (1999)
3. Franke, R., Nielson, G.M.: Scattered data interpolation and applications: a tutorial and survey. In: Geometric Modeling, pp. 131–160. Springer, Berlin (1991)
4. Thorsten, L., Michael, H., Wuensche, H.-J.: Autonomous ground vehicles concepts and a path to the future. Proc. IEEE **100**(13), 1831–1839 (2012)
5. Achtelik, M., Achtelik, M., Brunet, Y., Chli, M., Chatzichristofis, S.A.., Decotignie, J-D., Doth, K-M., Fraundorfer, F., Kneip, L., Gurdan, D., Heng, L., Kosmatopoulos, E.B., Doitsidis, L., Lee, G.H., Lynen, S., Martinelli, A., Meier, L., Pollefeys, M., Piguet, D., Renzaglia, A., Scaramuzza, D., Siegwart, R., Stumpf, J., Tanskanen, P., Troiani, C., Weiss. S.: Sfly: swarm of micro flying robots. In: IROS, pp. 2649–2650. IEEE (2012)
6. Birk, A., Pfingsthorn, M., Bülow, H.: Advances in underwater mapping and their application potential for safety, security, and rescue robotics. In: IEEE International Symposium on Safety, Security, Rescue Robotics (SSRR). IEEE Press (2012)
7. Michael, N., Shaojie, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., Tadokoro, S.: Collaborative mapping of an earthquake-damaged building via ground and aerial robots. J. Field Robot. **29**(5), 832–841 (2012)
8. Blosch, M., Weiss, S., Scaramuzza, D., Siegwart, R.: Vision based MAV navigation in unknown and unstructured environments. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 21–28. IEEE (2010)

9. Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor mav. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 4557–4564. IEEE (2012)
10. Majdik, A., Albers-Schoenberg, Y., Scaramuzza, D.: MAV urban localization from google street view data. In: IROS, pp. 3979–3986 (2013)
11. Doitsidis, L., Weiss, S., Renzaglia, A., Achtelik, M.W., Kosmatopoulos, E.B., Siegwart, R., Scaramuzza, D.: Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. Auton. Robots **33**(1–2), 173–178 (2012)
12. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: research challenges. Adhoc Netw. **3**(3), 257–279 (2005)
13. Kapoutsis, A.Ch., Chatzichristofis, S.A., Doitsidis, L., Borges de Sousa, J., Kosmatopoulos, E.B.: Autonomous navigation of teams of unmanned aerial or underwater vehicles for exploration of unknown static & dynamic environments. In: 21st Mediterranean Conference on Control & Automation (MED), 2013, pp. 1181–1188. IEEE (2013)
14. Bohling, G.: Introduction to Geostatistics and Variogram Analysis, p. 20. Kansas Geological Survey, Kansas (2005)
15. Ripley, B.D.: Spatial Statistics, vol. 575. Wiley.com, New York (2005)
16. Webster, R., Oliver, M.A.: Geostatistics for Environmental Scientists. Wiley, Chichester (2007)
17. Christopher, C.M., Condal, A.R.: A spatial data structure integrating GIS and simulation in a marine environment. Mar. Geodesy **18**(3), 213–228 (1995)
18. Burrough, P.A.: Principles of Geographical Information Systems for Land Resources Assessment (1986)
19. Charalampous, K., Amanatiadis, A., Gasteratos, A.: Efficient robot path planning in the presence of dynamically expanding obstacles. In: ACRI, volume 7495 of Lecture Notes in Computer Science, pp. 330–339. Springer (2012)
20. Ioannidis, K., Sirakoulis, GCh., Andreadis, I.: Cellular automata-based architecture for cooperative miniature robots. J. Cell. Autom. **8**(1–2), 91–111 (2013)
21. Chatzichristofis, S.A., Mitzias, D.A., Sirakoulis, GCh., Boutalis, Y.S.: A novel cellular automata based technique for visual multimedia content encryption. Opt. Commun. **283**(21), 4250–4260 (2010)
22. Zagoris, K., Pratikakis, I.: Scene text detection on images using cellular automata. In: ACRI, pp. 514–523 (2012)
23. Georgoudas, I.G., Sirakoulis, GCh., Scordilis, E.M., Andreadis, I.: A cellular automaton simulation tool for modelling seismicity in the region of Xanthi. Environ. Modell. Softw. **22**(10), 1455–1464 (2007)
24. Von Neumann, J., Burks, A.W., et al.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
25. Chopard, B., Droz, M.: Cellular Automata Modeling of Physical Systems. Cambridge University Press, New York (1998)
26. Toffoli, T.: Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. Phys. D Nonlinear Phenom. **10**(1–2), 117–127 (1984)
27. Bialynicki-Birula, I.: Weyl, Dirac, and Maxwell equations on a lattice as unitary cellular automata. Phys. Rev. D **49**(12), 6920–6927 (1994)
28. Omohundro, S.: Modelling cellular automata with partial differential equations. Phys. D Nonlinear Phenom. **10**(1–2), 128–134 (1984)
29. Malamud, B.D., Turcotte, D.L.: Cellular-automata models applied to natural hazards. Comput. Sci. Eng. **2**(3), 42–51 (2000)
30. Sirakoulis, GCh., Karafyllidis, I., Mardiris, V., Thanailakis, A.: Study of lithography profiles developed on non-planar Si surfaces. Nanotechnology **10**, 421–427 (1999)
31. Sirakoulis, GCh., Karafyllidis, I., Thanailakis, A.: A cellular automaton model for the effects of population movement and vaccination on epidemic propagation. Ecol. Modell. **133**(3), 209–223 (2000)

32. Sirakoulis, GCh., Karafyllidis, I., Thanailakis, A., Mardiris, V.: A methodology for VLSI implementation of cellular automata algorithms using VHDL. Adv. Eng. Softw. **32**(3), 189–202 (2000)
33. Sirakoulis, GCh., Karafyllidis, I., Thanailakis, A.: A CAD system for the construction and VLSI implementation of cellular automata algorithms using VHDL. Microprocess. Microsyst. **27**(8), 381–396 (2003)
34. Sirakoulis, GCh.: A TCAD system for VLSI implementation of the CVD process using VHDL. Integr. VLSI J. **37**(1), 63–81 (2004)
35. Mardiris, V., Sirakoulis, GCh., Mizas, Ch., Karafyllidis, I., Thanailakis, A.: A CAD system for modeling and simualtion of computer networks using cellular automata. IEEE Trans. SMC-Part C **38**(2), 1–12 (2008)
36. Glynn, J., de Moustier, C., Huff, L.: Survey operations and results using a Klein 5410 bathymetric sidescan sonar. In: US Hydro (2007)
37. Weiss, S., Achtelik, M., Kneip, L., Scaramuzza, D., Siegwart, R.: Intuitive 3D maps for MAV terrain exploration and obstacle avoidance. J. Intell. Robot. Syst. **61**(1–4), 473–493 (2011)

Chapter 11
On the Use of Cellular Automata in Vision-Based Robot Exploration

Lazaros Nalpantidis

Abstract Cellular Automata constitute a powerful tool to model spatial and temporal relations of complex discrete systems. Visual information, as captured by digital imaging sensors, can be efficiently processed by such techniques. Furthermore, robot exploration is commonly based on discrete metric occupancy grid representations of the environment. This chapter covers possible uses of Cellular Automata along the whole pipeline of vision-based robot exploration algorithms, and focuses on specific implementation examples of robotic systems with integrated CA-enhanced vision algorithms.

11.1 Introduction

Robot exploration of unknown environments is a very active topic among the autonomous robotics community. Robots are expected to navigate in such environments, localize themselves, and perform mapping. Computational resources onboard a mobile robot are limited, and have to be shared among many concurrent tasks. Thus, computationally efficient methods are favored instead of complex solutions, which makes the use of efficient and inherently parallelizable tools, such as Cellular Automata (CA), very appealing.

Naturally, 2D and 3D perception is of utter importance in robot exploration tasks and robots need sensors to perceive their environment and adapt their behavior accordingly. Cameras have been constantly gaining popularity due to their decreasing size and price, as well as due to their ability to provide very rich descriptions of the world at very high frame rates. Stereo vision systems are constantly one of the most commonly used ways to obtain depth information, other alternatives being laser-based [14], Time of Flight (ToF), and structured light sensors. Sensors that, apart from color images of the scene, can simultaneously provide depth measurements, i.e. the so called RGB-D sensors, have been steadily gaining popularity. The most

L. Nalpantidis (✉)
Department of Mechanical and Manufacturing Engineering, Aalborg University Copenhagen, Copenhagen, Denmark
e-mail: lanalpa@m-tech.aau.dk

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_11

iconic RGB-D sensor is the Microsoft Kinect that apart from the RGB information provides as an additional "channel' the depth; i.e. a grayscale image where each pixel value denotes the corresponding depth.

What is common about digital images, either RGB or RGB-D, is that their pixels constitute a lattice. As a result, CA have been applied successfully to image processing [1, 9, 12, 16] and dealt efficiently with image enhancement operations such as noise filtering [15]. However, noise filtering, image enhancement or edge extraction are just some of the applications where CA have provided efficient solutions. This chapter examines how CA can be incorporated in various stages of vision-based robot systems, improving results with only minimal computational overhead. More specifically, Sect. 11.2 deals with a CA-enhanced stereo correspondence algorithm, while Sect. 11.3 builds further upon that, examining a stereo vision-based simultaneous localization and mapping (SLAM) algorithm that uses CA to improve the generated occupancy map.

11.2 Stereo Vision

Robots need to perceive their 3D environments in order to plan their activities and execute them. Stereo vision can provide the means to perceive depth out of two images (Fig. 11.1).

CA can be used to make the stereo correspondence search more efficient and robust against noise. Rather than just a noise removal tool applied to the input images or the resulting depth maps, as discussed in the previous section, we have explored the tight integration of simple CA rules in the heart of the stereo correspondence process. More specifically, in [12] a CA-enhanced stereo algorithm is presented. Every stereo correspondence algorithm makes use of a matching cost function so as to establish correspondence between two pixels. The results of the matching cost computation comprise the disparity space image (DSI). DSI is a 3D matrix containing the computed matching costs for every pixel and for all its potential disparity values

Fig. 11.1 A robot equipped with a stereo camera, operating outdoors

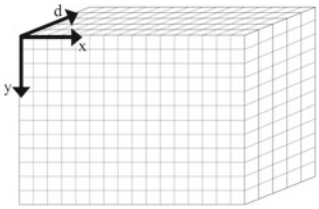


Fig. 11.2 DSI containing matching costs for every pixel of the image and for all its potential disparity values. Values within this 3D structure are processed with CA

[10]. The structure of a DSI is illustrated in Fig. 11.2. The algorithm presented in [12] utilizes the absolute differences (AD) as matching cost and aggregates the results inside support windows, assigning Gaussian distributed weights to the support pixels, based on their Euclidean distance. The resulting DSI is then refined by CA acting in all of the three dimensions of the DSI.

The main merit of the presented algorithm is its simplicity, rendering it as an ideal choice for real-time operations and hardware implementation. Its structural elements are summarized as:

1. AD is utilized as matching cost function since it is the simplest one, involving no multiplications.
2. The aggregation step is a 2D process performed inside fix-sized square support windows upon a slice of the DSI. The pixels inside each support window are assigned to a Gaussian distributed weight during aggregation. The weight of each pixel is a Gaussian function of its Euclidean distance towards the central pixel of the current window.
3. The resulting aggregated values of the DSI are furthered refined by applying CA. CA are used inside the 3D DSI, and not as a 2D post-processing disparity map filter [7].
4. Finally, the best disparity value for each pixel is decided by a WTA selection step.

This algorithm was not indented to achieve excellence of results but to provide a simple to implement, fast to execute yet credible stereo correspondence methodology. In this way the presented algorithm is able to be executed in real-time and to be easily hardware implemented, as demanded by many applications.

11.2.1 Algorithm Description

The main differentiation of the used algorithm from the majority of stereo algorithms is that the matching cost aggregation step consists of two sub-steps rather than one. In addition, the disparity selection process is a non-iterative one. Finally, the calculated disparity map is not further refined. The block diagram of the presented algorithm is shown in Fig. 11.3.

The key idea is that instead of refining the resulting 2D disparity map , refinement should be performed inside the 3D DSI. Thus, all the available information could

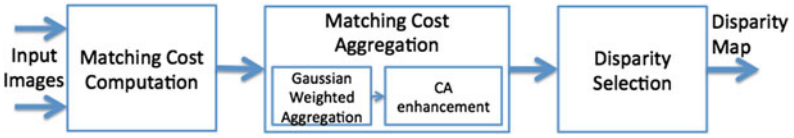


Fig. 11.3 Block diagram of the a CA-enhanced stereo correspondence algorithm (adapted from [12])

be taken into consideration. After all, WTA is a rigid information rejection method, often rejecting useful information as well. This alteration preserves the quality of the produced results while removing any iterative stage from the algorithm's flow.

The matching cost function utilized in the presented algorithm is the AD. The main merit of AD is the speed of calculations, since it involves only summations and finding absolute values, and its potential to be easily hardware implemented. Moreover, the results obtained by it are very satisfactory, considering the amount of avoided calculations that would be required by other metrics found in literature [10].

The AD calculated in the previous step comprise the DSI. These results are aggregated inside fix-sized square windows for constant value of disparity. The width of the window plays an important role on the final result. Small windows generally preserve details but suffer from noise, whereas big windows have the inverse behavior. After extensive testing to perform best, the width of the square window is selected to be 11 pixels. This number is considered a rational choice, as it manages to keep a balance between the loss of detail and the emergence of noise.

However, the AD summation is weighted. Each pixel is assigned a weight $w(i, j, d)$, the value of which results from the 2D Gaussian function of the pixel's Euclidean distance from the central pixel. The center of the function coincides with the central pixel and has a standard deviation equal to the one third of the distance from the central pixel to the nearest window-border. The Gaussian weight function remains the same for fixed width of the support window. Thus, it can be considered as a fixed mask that can be computed once, and then applied to all the windows.

The weighted SAD comprises the DSI:

$$DSI(i, j, d) = \sum_{\mu=-5}^{\mu=5} \sum_{\nu=-5}^{5} w(i+\mu, j+\nu, d) \cdot AD(i+\mu, j+\nu, d) \qquad (11.1)$$

The resulting aggregated values of the DSI are furthered refined by applying CA. All cells can work in parallel and as a result the used CA can be easily implemented in hardware. Two CA transition rules are applied to the DSI. The values of parameters used by them were determined after extensive testing to perform best. The first rule attempts to resolve disparity ambiguities. It checks for excessive consistency of results along the disparity d axis and, if necessary, corrects on the perpendicular (i, j) plane. The second rule is placed in order to smoothen the results and at the

same time to preserve the details. It checks and acts on constant-disparity planes. The two rules can be expressed as:

1. If at least one of the two pixels lying from either sides of a pixel across the disparity axis d differs from the central pixel less than half of its value, then its value is further aggregated within its 3×3 pixel, constant-disparity neighborhood.

   First CA rule Pseudocode

```
if {
   |DSI(i,j,d)-DSI(i,j,d-1)| < (1/2)DSI(i,j,d) }
or {
   |DSI(i,j,d)-DSI(i,j,d+1)| < (1/2)DSI(i,j,d) }
then {
   for m,n = (-1,0,1) {
      DSI(i,j,d) = (1/9)sum(sum(DSI(i+m,j+n,d) }}
```

2. If there are at least 7 pixels in the 3×3 pixel neighborhood which differ from the central pixel less than half of the central pixel's value, then the central pixel's value is scaled down by the factor 1.3, as dictated by exhaustive testing.

   Second CA rule Pseudocode

```
for m,n = (-1,0,1) {
   while (m and n)<>0 {
      if {
         |DSI(i+m,j+n,d)-DSI(i,j,d)| < (1/2)DSI(i,j,d) }
      then {
         count++ }}}
if {
   count>=7 }
then {
   DSI(x,y,d) = (1/1.3)DSI(i,j,d) }
```

The two rules are applied once. Their outcome comprises the enhanced DSI that will be used in order the optimum disparity map to be chosen by a simple, non-iterative WTA final step.

In the last stage the best disparity value for each pixel is decided by a WTA selection procedure. For each image pixel coordinates (i, j) the smaller value is searched for on the d axis and its position is declared to be the pixel's disparity value. That is:

$$D(i, j) = \arg(\min(DSI(i, j, d))) \tag{11.2}$$

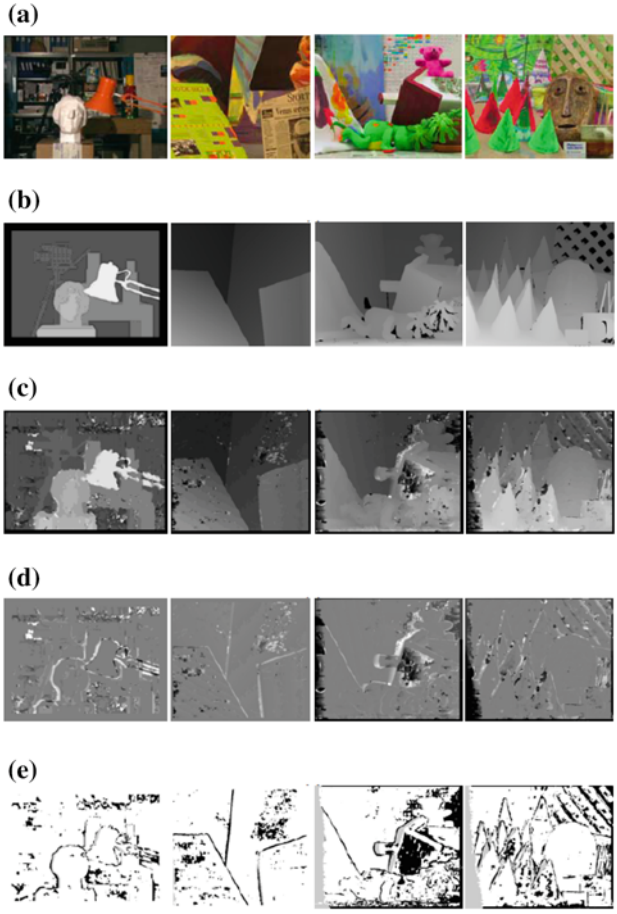


Fig. 11.4 Results for the Middlebury data sets. From *left* to *right*: the Tsukuba, Venus, Teddy and Cones image From *top* to *bottom*: the reference (*left*) images (**a**), the provided ground truth disparity maps (**b**), the disparity maps calculated by the presented method (**c**), maps of signed disparity error (**d**), and maps of pixels with absolute computed disparity error bigger than one (**e**)

11.2.2 Experimental Evaluation

The standard image sets used were the four stereo images [17, 18] provided along with their corresponding ground truth disparity maps by Scharstein and Szeliski through their web site [19]. Figure 11.4 depicts the reference (left) images (a), the provided ground truth disparity maps (b), the disparity maps calculated by the presented method (c), maps of signed disparity error where middle (50 %) gray tone equals to zero error (d), and maps of pixels with absolute computed disparity error bigger than one shown in black (e). The percentage of pixels whose absolute disparity error is greater than one in the non-occluded, all, and near discontinuities

and occluded regions are presented in Fig. 11.5. The presented algorithm leaves non-calculated a frame around the image whose width is equal to the aggregation window width, i.e. 11 pixels. Thus, the results of Fig. 11.5 slightly underestimate the performance of the presented algorithm, except for the case of Tsukuba image set, where the ground truth itself ignores that frame as well.

Figure 11.6, on the other hand, presents the Normalized Mean Square Error (NMSE) for the calculated disparity maps of the four image sets, excluding the 11 pixel wide frame. NMSE is calculated for a simplified version of the presented algorithm, which makes no use of CA, as well as for the complete version of the algorithm. The addition of CA substantially improves the quality, as shown from the last column.

11.2.3 Discussion

The presented algorithm exhibits satisfactory performance despite its simple structure. Gaussian weighted aggregation and CA refinement inside the DSI have been proven to comprise an effective computational combination. The data show that the presented algorithm is in the right direction for a hardware implementable, real-time solution. However, the quality of the results could be further improved by refining further the applied CA rules. The possibilities concerning the nature and the number of the applied CA rules are practically endless and the chosen ones, although effective, are only one of those possibilities. The presented algorithm's ability to calculate disparity maps of real-life scenes is highly appreciated. Finally, it can be concluded that the algorithm's serial flow and low complexity combined with the presented satisfactory results render it as an appealing candidate for hardware implementation. Thus, depth calculation could be performed efficiently in real-time by autonomous robotic systems.

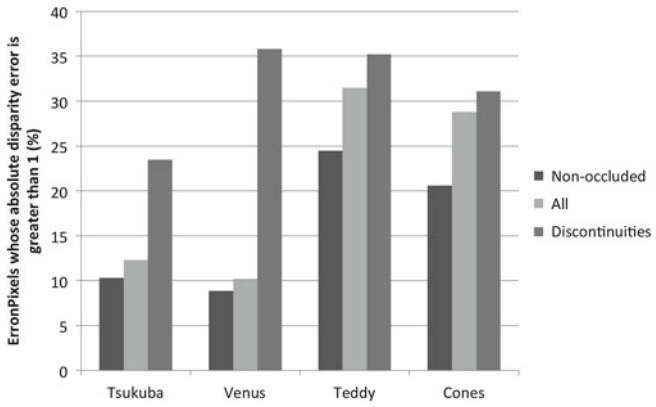


Fig. 11.5 Percentage of pixels whose absolute disparity error is greater than one in various regions of the images

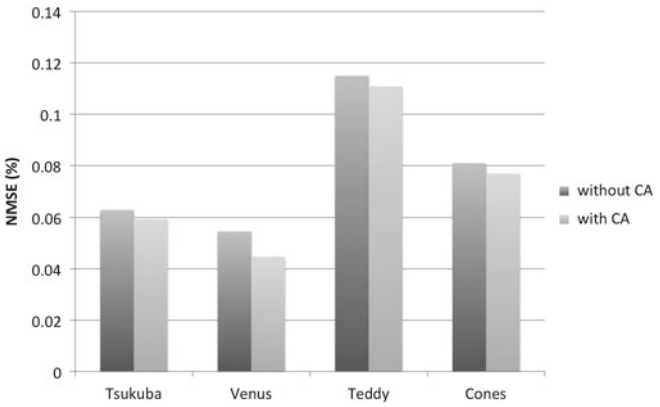


Fig. 11.6 Calculated NMSE on the Middlebury dataset for the presented stereo algorithm with and without CA enhancement

11.3 CA Refinement of Simultaneous Localization and Mapping

The use of CA in the SLAM problem has been explored in [11, 13]. SLAM is about estimating the robot's position and progressively building a map of its environment.

The difficulties of solving SLAM arise from the finite precision of the sensors and actuators of the robot, given real-life situations. The computational load of SLAM is also an additional problem. Much effort has been devoted to reduce the demanded computations [2, 6]. Yet, updating maps after each new observation requires more resources as the maps are getting larger. A typical example is the SLAM methods based on particle filters, where the selected number of particles and the consequent map matching and merging procedures significantly affect the computational load [2]. On the other hand, a simpler solution has been adopted in the method at hand. Instead of iteratively updating a large number of estimations, the presented method uses a corrective jittering procedure which aligns the most recently produced local map with the accumulated global map so as to minimize the occupied area of the resulting new global map. Then, carefully chosen CA rules are used to refine the resulting occupancy map. Thus, possible mistakes in the camera's motion estimation or mistakenly merged previous local maps can be overcome.

The solution presented in this method avoids complex update strategies in favor of a computationally efficient one. The only sensor required by the presented algorithm is a stereo camera. The resulting maps indicate the occupied regions of the area and, thus, can be used for area measurement applications. Emphasis has been given to the development of custom-tailored, non-iterative solutions for each step of the proposed algorithm's execution. The specially developed stereo correspondence algorithm is a rapidly executed local algorithm embodying gaussian weighted aggregation as well as a double validation scheme based on a certainty estimation criterion and a bidirectional consistency check. Concerning the camera's motion estimation, the

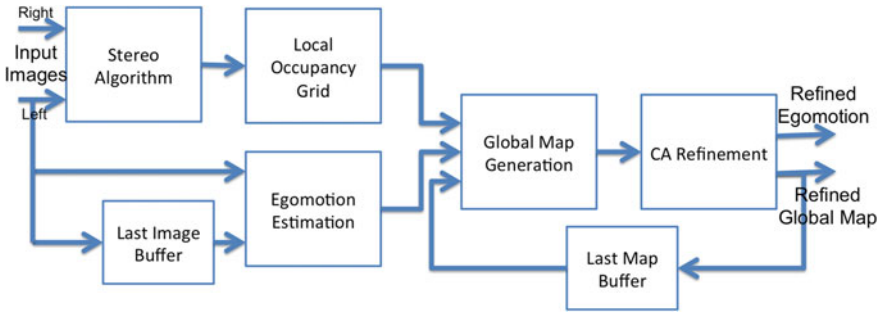


Fig. 11.7 Outline of the CA-enhanced SLAM algorithm (adapted from [13])

Speeded-Up Robust Features (SURF) feature detector and matcher [3] has been utilized as the first step of an efficient estimation method. This estimation is further refined afterwards during a sophisticated map merging procedure and sharpened up by CA [24].

11.3.1 SLAM Algorithm Description

The algorithm presented in [13] progressively builds a map of the environment, based entirely on stereo vision information. The produced maps indicate the occupied and free regions of the explored environment. The outline of the algorithm is summarized in Fig. 11.7. Each independent component is discussed in detail in the following sections.

11.3.1.1 Egomotion Estimation

The egomotion of the robot can be estimated by correlating the reference images from two consecutive image pairs. Feature detection and matching has become a very attractive and useful field for many computer vision applications. Among the variety of possible detectors and descriptors this work has embodied SURF, as described in [3]. SURF is a scale and rotation invariant detector and descriptor. It has the advantages of achieving high repeatability, distinctiveness and robustness. However, the most attractive feature of SURF is its computational efficiency, which allows very fast computation times. Preliminary experiments have confirmed the accuracy and effectiveness of SURF for the examined situations. SURF is given with two consecutive reference images as input and provides as output a list containing the coordinates of N matched features in the two images.

The coordinates of the pixels in each image are given in respect to that image's plain reference system. If the disparity values of the pixels matched by SURF have

been validly computed in the stereo correspondence step, then the local reference system's coordinates of the depicted point, i.e. with respect to the current camera position, can be computed by triangulation, given the intrinsic parameters of the stereo camera. Thus, the 3D coordinates, or equivalently two 3D point clouds, of the matched points for each stereo image pair are obtained as output.

The assumption required in the analysis hereafter is that the observed environment is static concerning the two adjacent image pairs and that the only non static object is the camera. However, even a partially non-static environment can be successfully processed most of the times. Consider for example the case of a moving person inside the room being mapped. The existence of the person will result in unwanted marks of obstacles. However, these "obstacles" will normally be present in a given position only for one single image pair, as the person will have moved till the capture of the next image pair. As a result, such a moving obstacle will produce weak obstacle traces, i.e. found only in a few of the input image pairs. The CA mechanism that will reject these non-existent obstacles is a subsequent step of the algorithm, which will be discussed later.

Consider the resulting two 3D point clouds. In this case the local coordinates of the features' position vectors $p'_1, \ldots, p'_N$ in the reference image of the second pair are related to the position vectors $p_1, \ldots, p_N$ in the reference image of the first pair by the equation:

$$p_i = T + R \cdot p'_i \qquad \text{for i} = 1, 2, \ldots, \text{N} \tag{11.3}$$

where T and R are the translation and the rotation matrices respectively, describing the camera's movement between the two reference images. In the ideal case, six perfectly matching features are sufficient in order to compute T and R. However, in realistic, error-suffering situations more points are needed. The sought T and R should minimise the following sum of quadratic differences [21]:

$$\sum_{i=1}^{N} ||p_i - T - R \cdot p'_i||^2 \tag{11.4}$$

The minimisation of this equation exploiting linear algebra, i.e. the application of a Procrustes transformation to the resulting two point clouds, results in the relative translation $T(x_0, y_0)$ and rotation $R(\theta_0)$ of the rover. This way, a linear transformation is determined between the points of the first point cloud and the points of the second one.

11.3.1.2 Local Map Generation

A local 2D map is computed from each stereo image pair. Using the sparse disparity map obtained by the stereo correspondence algorithm a reliable v-disparity image can be computed [8, 26]. In a v-disparity image each pixel has a positive integer

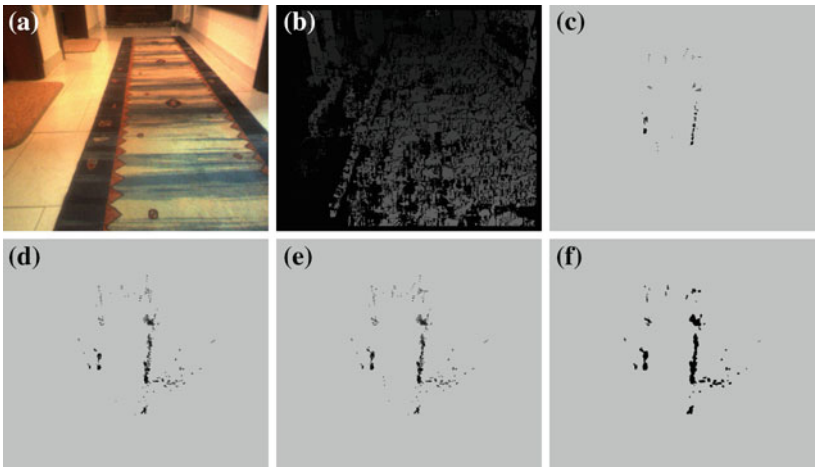


Fig. 11.8 Environment's maps for a domestic scene obtained with the presented method and enhanced using CA. **a** Observed hallway, **b** Computed disparity map, **c** Local map, **d** Initial global map, **e** Updated global map, **f** CA enhanced global map

value that denotes the number of pixels in the input image that lie on the same image line (ordinate) and have disparity value equal to its abscissa [8].

The terrain in the v-disparity image is mapped into a slanted line segment, which can be modeled by a linear equation. The parameters of this linear equation contain the information about the height and the angle of the camera with respect to the ground. The values of the linear equation's parameters can be found using Hough transform [4] or by exploiting an inertial measurement unit (IMU), if the camera-environment system's geometry is unknown. However, if the geometry of the system is constant and known (which is the case for a camera firmly mounted on a robot exploring a flat, e.g. indoor, environment) the two parameters can be easily computed beforehand and used in all the image pairs during the exploration. A tolerance region on either side of the terrain's linear segment is considered and any point outside this region is considered as an "obstacle". For each pixel corresponding to an "obstacle" the local coordinates are computed. The local map, e.g. the one shown in Fig. 11.8c, is an occupancy grid of the environment consisting of all the points corresponding to "obstacles".

11.3.1.3 Global Map Merging

The motion estimation technique, as described in the respective section, gives the relative translation $T(x_0, y_0)$ and rotation $R(\theta_0)$ required to superimpose the new local map, Fig. 11.8c, to the global map accumulated up to that point, Fig. 11.8d. However, the situation of perfectly matched features that result in exactly precise T

and R is barely ever encountered. Thus, one further step is required to address this issue.

The concept of aligning clouds of robust features is not a new one [20]. Stochastic and probabilistic methods have also been examined [5, 22] giving accurate results, at the expense of computational load. The solution adopted in this work is to accept the translation and rotation vectors $T(x_0, y_0)$ and $R(\theta_0)$ only as a good starting point. The fine tuning of the new local map with reference to the global one can be performed as a jittering procedure around the initial estimation. More specifically, the values of the translation and rotation vectors are iteratively updated so as to obtain a combination of values $x_1 \in [x_0 - 5, x_0 + 5]$, $y_1 \in [y_0 - 5, y_0 + 5]$ and $\theta_1 \in [\theta_0 - 4, \theta_0 + 4]$ (where x_0 and y_0 are measured in pixels and θ_0 in degrees) that when applied to the local map before the superimposition minimizes the total occupied area of the resulting new global occupancy map. This combination is considered as the fine-tuned values which determine the final T and R vectors and are used for the generation of the updated global map, e.g. the one shown in Fig. 11.8e.

11.3.1.4 Global Map Refinement Using CA

The algorithm's final output is a global map, i.e. an occupancy grid map of the explored area, whenever the user asks for it or when a termination criterion is met. The results of Fig. 11.8 are based on a series of self-captured stereo pairs. As shown, the global map usually needs to be filtered in order to fill possible gaps in it or remove unwanted noise. Such an enhancement is difficult to be achieved by typical image filtering techniques. This work proposes the use of planar CA for the refinement step, as they can easily deal with local interactions and can significantly improve the quality of the final result, as shown in Fig. 11.8f.

Despite its ease of implementation and simplicity, CA is a powerful tool that can generate robust filtering techniques. In their original definition, CA are dynamical systems, where space and time are discrete, interactions are local and they can easily handle complicated boundary and initial conditions [24, 25]. Using a more mathematical definition, a CA system requires:

1. a regular lattice of cells covering a portion of a d-dimensional space;
2. a set $C(\mathbf{r}, t) = \{C_1(\mathbf{r}, t), C_2(\mathbf{r}, t), \ldots, C_m(\mathbf{r}, t)\}$ of variables attached to each position $\mathbf{r}$ of the lattice giving the local state of each cell at the time $t = 0, 1, 2, \ldots$;
3. a rule $\boldsymbol{R} = \{R_1, R_2, \ldots, R_m\}$ which specifies the time evolution of the states $C(\mathbf{r}, t)$ in the following way:

$$C_j(\mathbf{r}, t+1) = R_j\{\, C(\mathbf{r}, t), C(\mathbf{r} + \boldsymbol{\delta}_1, t), C(\mathbf{r} + \boldsymbol{\delta}_2, t) \ldots, C(\mathbf{r} + \boldsymbol{\delta}_q, t)\} \tag{11.5}$$

where $\mathbf{r} + \boldsymbol{\delta}_k$ designate the cells belonging to a given neighborhood of cell $\mathbf{r}$.

In the above definition, the rule $\boldsymbol{R}$ is identical for all sites and it is applied simultaneously to each of them, leading to a synchronous dynamics. It is important to notice that the rule is homogeneous, i.e. it does not depend explicitly on the cell position $\boldsymbol{r}$.

The neighborhood of cell $\boldsymbol{r}$ is the spatial region in which a cell needs to search in its vicinity. In principle, there is no restriction on the size of the neighborhood, except that it is the same for all cells. However, in practice, it is often made up of adjacent cells only. If the neighborhood of each cell is defined by the variable N, then for a 2D CA, two neighborhoods are often considered, Von Neumann and Moore neighborhood. Von Neumann neighborhood is a diamond shaped neighborhood and can be used to define a set of cells surrounding a given cell, $\boldsymbol{r}$ with given coordinates (i_0, j_0). The following equation defines the Von Neumann neighborhood, N^{von}, of range p:

$$N(i_0, j_0)^{von} = \{(i, j) : |i - i_0| + |j - j_0| \leq (p)\} \tag{11.6}$$

For a given cell $\boldsymbol{r}$ with given coordinates (i_0, j_0) and range of neighborhood p, the Moore neighborhood, a square shaped neighborhood can be defined by the following equation, respectively:

$$N_(i_0, j_0)^{Moore} = \{(i, j) : |i - i_0| \leq (p), |j - j_0| \leq (p)\} \tag{11.7}$$

The CA rules as mentioned above determine the way in which each cell of the CA is updated. It is clear that the state of each cell is affected by the cell values in its neighborhood and its value on the previous time step, according to the transition CA rule, or a set of rules. The state of every cell is updated simultaneously in the CA, thus, providing an inherent parallel system. In practice, when simulating a given CA rule, it is impossible to deal with an infinite lattice. The system must be finite and have boundaries. Clearly, a site belonging to the lattice boundary does not have the same neighborhood as other internal sites. In order to define the behavior of these sites, the neighborhood is extending for the sites at the boundary leading to various types of boundary conditions such as periodic (or cyclic), fixed, adiabatic or reflection.

Based on the above CA presentation, the following two 2D majority rules are taken into account for the occupancy grid enhancement. It should be noticed that different neighborhood sizes were tested several times and for different images in order to provide more accurate results in correspondence to the requested computational resources. As a result, the extended CA Moore neighborhood with range $p = 2$ was finally selected as a compromise between accuracy and the corresponding computation time. Thus, considering $C_{o,o}$ as the central pixel of the 5×5 neighborhood, the used majority rules can be expressed by the following if-then statements:

$$\text{if } \sum_{i=-2}^{2} \sum_{j=-2}^{2} C_{i,j} < 4 \quad \forall\, (i \neq 0 \quad \text{and} \quad j \neq 0) \quad \text{then} \quad C_{o,o} = 0 \tag{11.8}$$

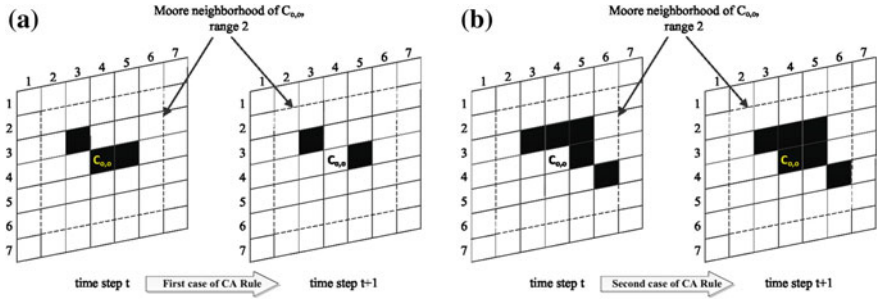


Fig. 11.9 Application of CA majority rules for the time evolution of the CA state $C_{o,o}$ for different initial conditions and for Moore neighborhood with range $p = 2$. The CA states of the *black pixels* equal to 1 while the CA states of the *white pixels* equal to 0

and

$$\text{if} \quad \sum_{i=-2}^{2} \sum_{j=-2}^{2} C_{i,j} > 5 \quad \forall \quad (i \neq 0 \quad \text{and} \quad j \neq 0) \quad \text{then} \quad C_{o,o} = 1 \tag{11.9}$$

In Fig. 11.9, two application examples of the aforementioned CA majority rules are provided, respectively. More specifically, in Fig. 11.9a the CA cell state of $C_{o,o}$ results in next time $t + 1$ in 0, according to the first majority rule in the above mathematical description, since the sum of all its neighbors states equals to 2; while in Fig. 11.9b the CA cell state of $C_{o,o}$ results in next time $t + 1$ in 1, according to the second majority rule, since the sum of all its neighbours states equals to 5. In both the previous examples, it should be mentioned that, for readability reasons, all the CA cell states outside the neighborhood of the under study CA cell $C_{o,o}$ are not taken into account and as a result the states of all other CA cells depicted in Fig. 11.9a, b are not evaluated. The aforementioned CA rules were explicitly implemented after extensive testing to perform best, according to the accuracy speed trade-off of the method. Finally, the boundary conditions of the used CA were selected to be fixed zero-valued for better computational results.

Just one generation, i.e. a single application of the aforementioned CA rules, provides significantly enhanced results, as seen in Fig. 11.8f. The barely connected 2D point cloud of Fig. 11.8e has been transformed to the solid occupied region shown in Fig. 11.8f. Due to the fact that the presented global map enhancement uses a CA, time complexity is found to be approximately $O(m \times n)$ as the majority of the CA algorithms, where $m \times n$ are the dimensions of the rectangular grid, meaning that time complexity is proportional to the CA dimensions. The main advantages of the presented solution are its significantly enhanced results, combined with low complexity and low computational cost, thus, suitable for an efficient real-time system implementation.

Fig. 11.10 Robot equipped with Bumblebee stereo camera used in experiments

11.3.2 Experimental Evaluation

The presented algorithm was tested experimentally by being applied to a series of self-captured indoor images. Two sets of images were used for testing. The first set of images consisted of 10 stereo image pairs captured in a domestic indoor environment, i.e. a hallway with some open doors alongside. The second set of images constituted of 113 image pairs captured by a stereo camera-equipped robot exploring a public area of a university building. The used mobile robot can be seen in Fig. 11.10; it actually has two stereo cameras but the one that was used in the following experiments was the one placed lower—the yellow Point Grey Bumblebee2.

For the first set of images, i.e. for the images of the hallway, the orientation of the camera was only slightly different among successive captures, in order to facilitate the task of the feature matching procedure. The lighting of the environment was the natural lighting found in real indoor environments. As a result, the captured images were far from being ideal, obstructing the stereo correspondence and the feature matching procedures.

In Fig. 11.11 the first column, Fig. 11.11a, presents the reference images of the first, second, sixth, and tenth image pair of the tested image series. The differences in the illumination conditions are evident, especially in the image of the third row. The second column, Fig. 11.11b, presents the sparse disparity maps computed with the used stereo algorithm. One can observe that very little falsely matched pixels have been produced. However, the overall coverage is more than enough in order to to detect any obstacles in the scene. The third column, Fig. 11.11c, shows the global maps of the environment containing the accumulated local maps up to that point. The gradual superimposition of further local maps is remarkably accurate and results in clear arrangements of “obstacle”-points. The final column, Fig. 11.11d,

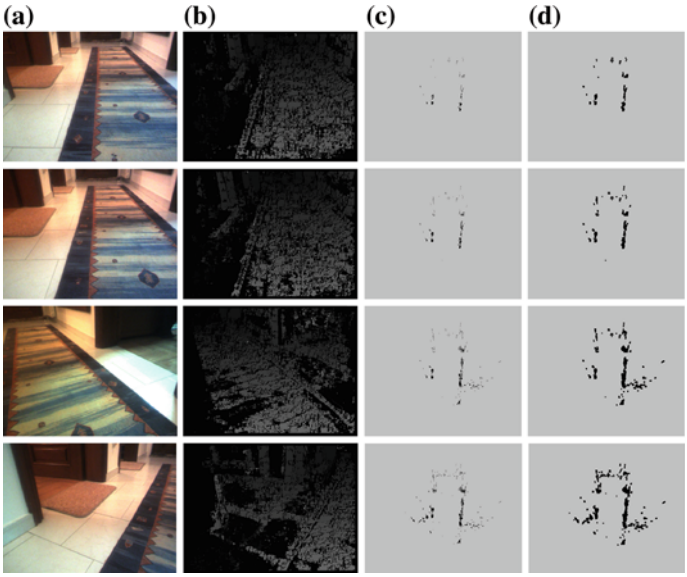


Fig. 11.11 Experimental results after processing 1 (*first row*), 2 (*second row*), 6 (*third row*), and 10 (*fourth row*) image pairs of the hallway scene. **a** Reference images of the scene. **b** Sparse disparity maps of the scene. **c** Updated global maps. **d** CA enhanced global maps

shows the global maps after the CA enhancement. This procedure, makes the sparse information of the global maps continuous and more clear.

As shown by the lower-right image of Fig. 11.11 the result of the algorithm with only ten input image pairs is a clear and reliable representation of the obstacles found in the hallway. The walls, the closed door straight ahead of the camera, and some open doors are all clearly visible.

The second set of images consisted of 113 stereo image pairs of a scene captured in a university building. The images were captured by a robot whose path resulted in a full closed loop inside the explored environment. The experimental results, presented in Fig. 11.12, shows that the presented methodology can handle large sequences of input images recovering from and counterbalancing the obviously error-including pose and mapping estimations.

The final CA enhanced maps for the two datasets, i.e. the hallway and the university ones, are shown in Fig. 11.13. Apart form the SLAM output, the outline of the mapped environment is superimposed in red for the purpose of comparison.

The accuracy of the produced maps depends on two different error inducing mechanisms, i.e. false matching and estimation inaccuracy. False matching is almost inevitable for any stereo correspondence algorithm to some extent. Such errors are handled by validation techniques. The disparity estimation inaccuracy occurs when the integer-pixel disparity value is found correctly but the subpixel precision value is erroneous. This case has a lot to do with the used camera's resolution. The camera's

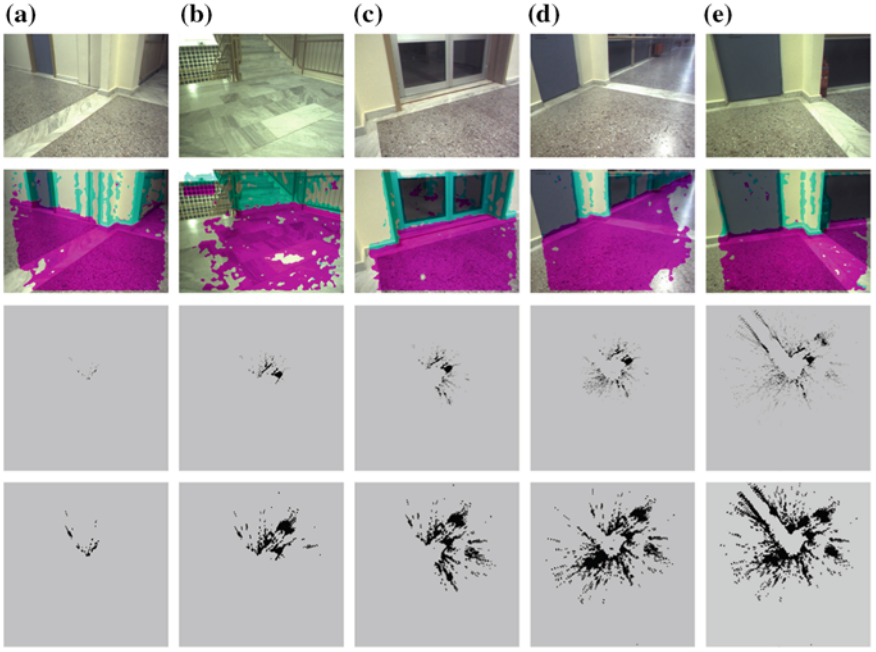


Fig. 11.12 From *top* to *bottom* the reference image, highlighted traversable areas and obstacles, global map, and CA enhanced global map for the **a** 1st, **b** 20th, **c** 50th, **d** 80th, and **e** 113th pair of the university scene

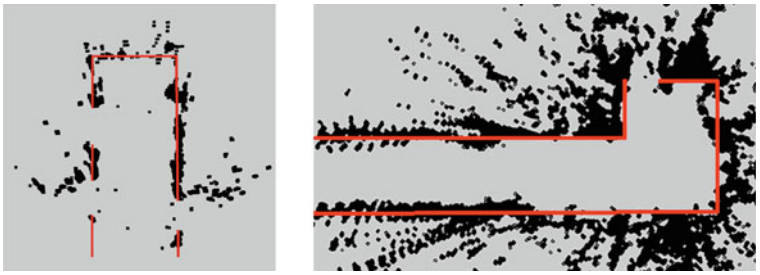


Fig. 11.13 The final CA enhanced global maps and the outline of the mapped environments

lenses had 3.8 mm focal length, resulting in 65° horizontal field of view (HFOV). One can derive that the used stereo vision system can produce maps of its environment with theoretical accuracy better than 0.85 %, for objects within a radial distance of up to 2 m.

11.3.3 Discussion

SLAM methods are able to produce maps of areas and provide metric results about occupied and free regions. However, SLAM is known to be a computational intensive task. A simple and computationally efficient SLAM algorithm would comprise a useful tool for distance and area measurement applications. In this work, a computationally simple stereo vision-based SLAM algorithm has been presented that uses CA to produce accurate occupancy maps. New methodologies for the generation and the superimposition of the partial maps have been proposed. The sole use of one sensor, i.e. a stereo camera, and the substitution of computationally demanding procedures are indicative of the algorithm's focus on computational effectiveness.

The stereo correspondence algorithm used in this work is a rapidly executed local algorithm embodying gaussian weighted aggregation and a double validation scheme based on a certainty estimation criterion and a bidirectional consistency check.

The vision-based estimation of the camera's motion is based on the SURF feature detector and matcher in order to obtain a first rough estimation. This first estimation is further refined afterwards during a sophisticated and efficient map merging procedure. The superimposition of the latest local map to the accumulated global one is performed through a jittering procedure around the SURF-based estimation, which minimizes the overall occupied area of the map. Thus, for slightly differentiated consecutive image pairs a very good fitting can be achieved. In contrast to stochastic and probabilistic methods proposed in the relevant literature, the presented method has been found able to produce accurate results having only a small computational footprint.

The generally sparse results of the map merging are sharpened up by the use of CA. CA have been proved to be an effective tool for to image processing and image enhancement operations such as noise filtering. In this work, the application of proper CA majority rules accomplishes to remove unwanted noise and at the same time produce a continuous contour of the obstacles surface.

Finally, another interesting prospective concerning the hardware implementation of the presented stereo vision-based SLAM algorithm arrives from the suitability of the CA module itself used in the presented framework. CA design is most efficient when implemented in hardware, due to the highly parallel independent processing. More specifically, the reasons why CA can be ideally implemented by VLSI techniques are that from circuit designing point of view, according to Toffoli [23], there are four main factors that determine the cost/performance ratio of an integrated circuit, namely, circuit design and layout, ease of mask generation, silicon-area utilization, and maximization of achievable clock speed; for a given technology, the latter is inversely proportional to the maximum length of the signal paths. CA circuit design reduces to the design of a single, relatively simple cell with uniform layout. The whole mask for a large CA array (the cells with their internal connections as well as the interconnection between cells) can be generated by a repetitive procedure so no circuit area is wasted on long interconnection lines and because of the locality of processing, the length of critical paths is minimal and independent of the num-

ber of cells. Furthermore, based on the above circuit CA design characteristics the FPGA implementation of the corresponding CA algorithms is also straightforward. Beyond its simplicity the resulting hardware would present the advantages of parallel implementation with maximum speed.

11.4 Conclusion

Robots need to operate in their environments, i.e. sense, perceive and act, in a natural way. Vision-based perception is known to be computationally demanding. To ensure a natural flow of operation, without requiring stops for processing, one needs to find a balance between required accuracy and speed. This fact makes computational simple tools, such as CA, a very appealing choice for improving the results of simple (but fast) algorithms.

Visual images, either color or also depth ones, constitute a regular spatial lattice, while video streams add an additional, temporal, dimension to that. Furthermore, many data structures, used in vision algorithms for robot exploration have the form of regular latices, where locality plays a significant role in the stored information. In this chapter we have examined how CA can be incorporated in various stages of a stereo vision and a SLAM algorithm used in vision-equipped mobile robot systems. The results of both algorithms significantly improved, through the use of carefully designed CA rules, while the added computational overhead was very limited. These findings, together with the inherent nature of CA that allows for parallelization and very efficient hardware implementation, indicate that CA can be integrated even further and further improve the results and efficiency of mobile robotic systems.

References

1. Alvarez, G., Hernández Encinas, L., Martín del Rey, A.: A multisecret sharing scheme for color images based on cellular automata. Inf. Sci. **178**(22), 4382–4395 (2008)
2. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM). Part II. IEEE Robot. Autom. Mag. **13**(3), 108–117 (2006)
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**, 346–359 (2008)
4. De Cubber, G., Nalpantidis, L., Sirakoulis, G.C., Gasteratos, A.: Intelligent robots need intelligent vision: visual 3D perception. In: IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance. Benicàssim, Spain (2008)
5. Durrant-Whyte, H., Bailey, T.: Simultaneous localisation and mapping (SLAM). Part I the essential algorithms. IEEE Robot. Autom. Mag. **2**, 2006 (2006)
6. Huang, S., Wang, Z., Dissanayake, G.: Sparse local submap joining filter for building large-scale maps. IEEE Trans. Robot. **24**(5), 1121–1130 (2008)
7. Kotoulas, L., Gasteratos, A., Sirakoulis, G.C., Georgoulas, C., Andreadis, I.: Enhancement of fast acquired disparity maps using a 1-D cellular automation filter. IASTED International Conference on Visualization, Imaging and Image Processing, pp. 355–359. Benidorm, Spain (2005)

8. Labayrade, R., Aubert, D., Tarel, J.P.: Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In: IEEE Intelligent Vehicle Symposium, vol. 2, pp. 646–651. Versailles, France (2002)
9. Lafe, O.: Cellular Automata Transforms: Theory and Applications in Multimedia Compression, Encryption and Modeling. Multimedia Systems and Applications Series. Kluwer Academic Publishers, Norwell (2000)
10. Muhlmann, K., Maier, D., Hesser, J., Manner, R.: Calculating dense disparity maps from color stereo images, an efficient implementation. Int. J. Comput. Vision **47**(1–3), 79–88 (2002)
11. Nalpantidis, L., Sirakoulis, G.C., Carbone, A., Gasteratos, A.: Computationally effective stereovision SLAM. In: IEEE International Conference on Imaging Systems and Techniques, pp. 453–458. Thessaloniki, Greece (2010)
12. Nalpantidis, L., Sirakoulis, G.C., Gasteratos, A.: A dense stereo correspondence algorithm for hardware implementation with enhanced disparity selection. In: 5th Hellenic Conference on Artificial Intelligence. Lecture Notes in Computer Science, vol. 5138, pp. 365–370. Springer, Syros (2008)
13. Nalpantidis, L., Sirakoulis, G.C., Gasteratos, A.: Non-probabilistic cellular automata-enhanced stereo vision simultaneous localisation and mapping (SLAM). Measure. Sci. Technol. **22**(11), 114027 (2011)
14. Rekleitis, I., Bedwani, J.L., Dupuis, E., Lamarche, T., Allard, P.: Autonomous over-the-horizon navigation using lidar data. Auton. Robots **34**, 1–18 (2013)
15. Rosin, P.L.: Training cellular automata for image processing. IEEE Trans. Image Process. **15**, 2076–2087 (2006). doi:10.1109/TIP.2006.877040
16. Rosin, P.L.: Image processing using 3-state cellular automata. Comput. Vis. Image Underst. **114**, 790–802 (2010). http://dx.doi.org/10.1016/j.cviu.2010.02.005, http://dx.doi.org/10.1016/j.cviu.2010.02.005
17. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vision **47**(1–3), 7–42 (2002)
18. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. **1**, 195–202 (2003)
19. Scharstein, D., Szeliski, R.: http://vision.middlebury.edu/stereo/ (2010)
20. Se, S., Lowe, D., Little, J.: Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. Int. J. Robot. Res. **21**, 735–758 (2002)
21. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: Modelling Planning and Control. Springer Publishing Company, Incorporated, New York (2008)
22. Sim, R., Elinas, P., Little, J.: A study of the rao-blackwellised particle filter for efficient and accurate vision-based SLAM. Int. J. Comput. Vision **74**(3), 303–318 (2007)
23. Toffoli, T., Margolus, N.: Cellular Automata Machines: A New Environment for Modeling. MIT Press, Cambridge (1987)
24. Von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)
25. Wolfram, S.: Theory and Applications of Cellular Automata. World Scientific, Singapore (1986)
26. Zhao, J., Katupitiya, J., Ward, J.: Global correlation based ground plane estimation using v-disparity image. In: IEEE International Conference on Robotics and Automation, pp. 529–534. Rome, Italy (2007)

Chapter 12
Modelling Synchronisation in Multirobot Systems with Cellular Automata: Analysis of Update Methods and Topology Perturbations

Fernando Silva, Luís Correia and Anders Lyhne Christensen

Abstract In this chapter, we consider two-dimensional cellular automata as a tool for modelling the behaviour of multirobot systems. We study the dynamics of a group of stationary robots inspired by studies in mixed natural-artificial societies. We model the behaviour of individual robots as a pulse-coupled oscillator, which influences other oscillators through short and periodic pulses. We address the problem of self-organised synchronisation, in which robots have to adjust and synchronise their behaviour to produce a self-organised collective vibration pattern based on local interactions. We analyse under which conditions a synchronised global behaviour can emerge from local coupling between neighbours and focus on two fundamental aspects: (i) the effects of different update methods, including the interplay between parameters of local rules and the global behaviour, and (ii) the transition from regular to irregular topologies by means of dynamic perturbations and fixed perturbations. Overall, this study is a contribution towards the understanding of the effects of the update method and of the topological structure when modelling real-world complex systems with cellular automata.

12.1 Introduction

In this chapter, we study two-dimensional cellular automata as an abstraction for modelling and examining the behaviour of autonomous, mutually interacting sets of agents, i.e., multiagent systems. The key principles for modelling multiagent systems by means of discrete dynamical systems such as cellular automata have not

F. Silva (✉) · L. Correia
LabMAg, Faculdade de Ciências da Universidade de Lisboa, 1749-016 Lisbon, Portugal
e-mail: fsilva@di.fc.ul.pt

L. Correia
e-mail: luis.correia@di.fc.ul.pt

A.L. Christensen
Instituto de Telecomunicações, Instituto Universitário de Lisboa (ISCTE-IUL),
1649-026 Lisbon, Portugal
e-mail: anders.christensen@iscte.pt

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_12

yet been agreed upon [1]. Proposed approaches include: (i) translation of multiagent systems into corresponding cellular automata models [2], (ii) modification of the expressiveness and structure of cellular automata in order to provide a basis for direct modelling of groups of interacting agents [3], and (iii) hybrid approaches encompassing both cell-based updating and agent-based updating [4]. Influenced by the view of multiagent systems as discrete dynamical systems provided by Fatès and Chevrier [5], we define and model a multiagent system by separating three core concepts: (i) the formulation of individual behaviour, i.e., the local rules that describe the actions of the agents, (ii) the update method, which defines the temporal relation between modifications to the agents' internal state, and (iii) the topology of the environment, including connections between agents, which may change or remain fixed throughout time.

We study the collective dynamics of one class of multiagent systems: multirobot systems. The robots are modelled after the combined actuator-sensor units (CASUs), groups of stationary robots developed in the context of the ASSISIbf project [6] on mixed natural-artificial societies.[1] CASUs are designed according to the principles of distributed control and self-organisation. The robots are expected to coexist and interact with honeybees and zebrafish both at the individual level and at the group level. The CASUs are intended: (i) to modulate the behaviour of the animals, and (ii) to "learn the social language of the animals" [6].

In distributed systems, synchronisation plays a fundamental role [7]. Examples are ubiquitous and include: (i) imposing a reference timing in wireless networks [8], (ii) communication scheduling, coordinated duty cycling, and time synchronisation in sensor networks [9], and (iii) decentralised fault detection in groups of autonomous robots [10]. In our study, robots have to synchronise to produce self-organised collective behaviour. The robots are equipped with actuators to emit vibrations, and sensors that enable them to detect if neighbouring robots are vibrating. The behaviour of each robot is modelled as a pulse-coupled oscillator [11], which influences other oscillators through short and periodic pulses. Based on local interactions, robots have to adjust their behaviour in order to produce a common, population-wide vibration pattern. Our goal is to analyse and understand under which conditions a synchronised global behaviour can emerge from local coupling between neighbours. Our study is concerned with answering three fundamental questions:

- Since robots are not centrally synchronised, does the global self-synchronised behaviour of the multirobot system depend on the update method? Is the system sensitive or robust against changes in the updating? What is the most appropriate update method for modelling multirobot synchronisation of behaviour?
- Under a given update method, is the global behaviour influenced by variations of the parameters that regulate the individual behaviour of agents? In other words,

[1] ASSISIbf [6], short for "Animal and robot Societies Self-organise and Integrate by Social Interaction", is an EU-funded FP7 FET project. The project focuses on self-organising mixed societies of real social animals, namely bees and fish, and artificial systems. The ultimate goal of the project is to develop a new field of science concerning self-adapting engineered systems able to integrate themselves in existing natural societies.

what is the main cause of emergence of certain classes of behaviour: (i) specific configurations of parameters for the local rules, (ii) the update method, or (iii) the interplay between the configuration of the local rule parameters and the update method?
- How robust is the system to modifications in the topology? In particular, if the topology is perturbed by the removal of connections and therefore interactions between neighbouring agents, under which conditions can the system continue to exhibit the desired collective behaviour?

The chapter is organised as follows. Section 12.2 describes the background and related work. We present the cellular automata model, the update methods used, and we review relevant studies on pulse-coupled oscillators. In Sect. 12.3, we introduce a number of definitions and notations necessary to characterise the behaviour of cellular automata. In Sect. 12.4, we assess the impact of the update method on the evolution of synchronised behaviour. We also analyse the sensitivity of the update method to variations in key parameters of individual behavioural rules. In Sect. 12.5, we investigate the effects of topology perturbations. In particular, we analyse the robustness of cellular automata when topology characteristics become irregular due to the removal of connections between neighbouring cells. In Sect. 12.6, we summarise and discuss our contribution, and we present avenues for future research in modelling multirobot systems with cellular automata.

12.2 Background and Related Work

In this section, we first define the cellular automata model and we describe the five update methods studied in this chapter. Afterwards, we report the background and related work on oscillators, and we detail our pulse-coupled oscillator model.

12.2.1 Topology of the Environment

Let A be a two-dimensional squared automaton of length l with toroidal boundary conditions. We denote $N = l \times l$, the total number of cells, as the size of the automaton. Individual cells adopt one of two possible states in the set of states $S = \{0, 1\}$. The state of all cells at a given time is called a *configuration*.

The state of each cell is updated at discrete time steps according to a local transition function. The locality of the interactions between cells is given by the relation of each cell $c \in A$ with its *von Neumann* neighbourhood of radius $r = 1$: $N(c) = \{c, c + \overrightarrow{v}, c - \overrightarrow{v}, c + \overrightarrow{h}, c - \overrightarrow{h}\}$, where $\overrightarrow{v} = (0, r)$ and $\overrightarrow{h} = (r, 0)$. For a given cell c and an integer value k, we define the *field* $\varphi(c, k)$ as:

$$\varphi(c, k) = \{c' \in A \mid r(c, c') = k\}\ , \tag{12.1}$$

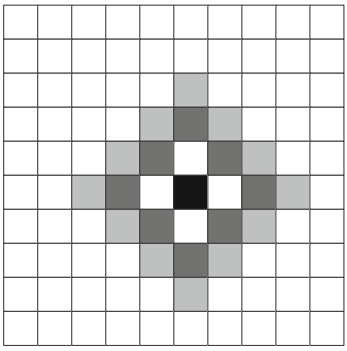


Fig. 12.1 Example of two fields, $\varphi(c, 2)$ and $\varphi(c, 3)$, for an automaton of size $N = 10 \times 10$. The centre cell c is marked in *black*. The fields $\varphi(c, 2)$ and $\varphi(c, 3)$ are marked in *dark grey* and *light grey*, respectively

where $r(c, c')$ is the radial distance between cells c and c'. In Fig. 12.1, we show an example of the fields $\varphi(c, 2)$ and $\varphi(c, 3)$ for an automaton of size $N = 10 \times 10$.

12.2.2 Update Methods

The update method determines the set of cells to which the local transition function applies and the order of the updating at each time step. In the classic definition, cellular automata are perfectly synchronous as cells are updated instantaneously and in parallel. The assumption of perfectly synchronous timing has been widely debated, with the main argument against being that synchrony presupposes the existence of a global clock that dictates the pace of all local processes in the system [12]. In addition, the synchronous update does not reflect the microscopic structure of physical and biological systems [13, 14]. Immediately after the issue of synchrony vs. asynchrony was raised, a number of additional research questions arose [15] including: Which properties of the system change when the update method is asynchronous instead of synchronous? What type of asynchrony should be used to model a given system?

A number of studies focusing on asynchronous update methods have been conducted, see for instance [16–22]. The contributions showed that the behaviour of cellular automata can change considerably when perfect synchrony is absent. Bersini and Detours [23], Ruxton and Saravia [24] and Cornforth et al. [13] argued that no single update method is suitable for all systems. Here, in order to understand which update method is more faithful to the physical reality of a given system, we study the effects of five different schemes:

1. Synchronous method, in which all cells are updated in parallel at each time step.
2. α-asynchronous method [19], which is based on a sampling scheme in which only a fraction of the cells is updated at each time step. Every cell has an independent and equal probability α of being updated, with $0 < \alpha < 1$, thus satisfying a fair sampling condition. The α-asynchronous method has been widely used to: (i) define robustness classes [19] and analyse behavioural changes under

asynchronous conditions in one-dimensional [25–28] and two-dimensional cellular automata [29–31], and (ii) to model multiagent systems in two-dimensional cellular automata [1, 32].

3. κ-scaling method [22], which extends standard α-asynchrony in order to compensate for fewer updates due to increasing asynchrony. The κ-scaling method is defined as follows: given a synchrony rate α, at each time step, $\kappa = 1/\alpha$ updates are performed. For non-integer $\kappa = 1/\alpha$, decimal values are probabilistically used for deciding between performing n or $n + 1$ intermediate updates. The resulting configuration is considered as the automaton's next stage. κ-scaling introduces a time-scaling phenomenon according to the synchrony rate, thus potentially normalising the differences introduced by α-asynchrony. In this way, the sampling compensation allows us to assess if differences in behaviour are due to less cell activity (asynchrony) or to more complex phenomena.
4. Random order asynchronous method in which, at each time step, all cells are updated exactly once according to a random order. Kanada [12], Schönfisch and de Roos [21] and Cornforth et al. [13] have previously assessed the effects of randomised update sequences in cellular automata. Randomness in the computational order was shown to significantly influence the spatiotemporal behavioural patterns. Depending on the local transition function, one-dimensional cellular automata were found to exhibit chaotic behaviour [12] and quasi-cyclic behaviour [13]. The random model update method has also been applied to cellular automata-based modelling of, for instance, biological systems [33] and chemical systems [34].
5. Line-by-line sweep, or fixed directional, is the simplest asynchronous method. All cells are updated at each time setup. The update is performed line-by-line, from the leftmost cell to the rightmost cell. The effects of this method have been compared with others by Rakewsky et al. [35] in the asymmetric simple exclusion process (ASEP), by Schönfisch and de Roos [21] in one-dimensional cellular automata, and by Ruxton and Saravia [24] in two-dimensional cellular automata-based ecological modelling. One of the conclusions of the studies was that the line-by-line sweep can introduce additional structure in the evolution of the automata. We use the method to analyse if such a cyclic behaviour can offer any benefit in modelling synchronisation of multirobot systems.

12.2.3 Modelling Individual Behaviour with Pulse-coupled Oscillators

Our study is concerned with modelling and studying the emergence of synchronised behaviour in a population of *stationary* robots. The behaviour of each individual robot is modelled as a pulse-coupled oscillator [11]. In the following sections, we describe the motivation for the task, the background on pulse-coupled oscillators, the robot model, and how the behavioural control of robots is defined.

12.2.3.1 Synchronisation of Pulse-coupled Oscillators

Self-organised synchronisation is a common and important phenomenon in natural systems. A number of natural collective systems are subject to a strong and regular synchronisation component, and they synchronise their behaviour based on local interactions, i.e., simple coupling rules at the individual level result in a consistently synchronised behaviour. Complete synchronisation of behaviour is thus a progressive and emergent phenomenon [36]. Examples of natural synchronisation phenomena include tropical fireflies that routinely synchronise their rhythmic flashes [37], cardiac cells [38], circadian pacemaker cells in the brain, metabolic synchrony in yeast cell suspensions, and crickets that chirp in unison [39].

The phenomenon of collective synchronisation in a number of natural systems is described by pulse-coupled oscillators that spontaneously lock into a common phase. Each oscillator periodically emits a self-generated pulse. When other oscillators observe the pulse, they slightly shift their own oscillation phase. This process leads to an alignment of phases and to synchronised behaviour. Let us consider the aforementioned tropical fireflies. At an individual level, fireflies have neural timing mechanisms, an oscillator. The only interaction occurs when fireflies flash, which stimulates or inhibits the oscillation frequency of neighbouring fireflies and causes them to modify their internal rhythm [11].

The first example of synchronisation of pulse-coupled oscillators was described by Peskin [40], who observed the phenomenon in the cardiac pacemaker cells. Afterwards, Mirollo and Strogatz [11] demonstrated that any number of pulse-coupled oscillators is always able to synchronise their firing rates as long as: (i) the population of oscillators is fully connected, i.e., each oscillator is coupled with all the others, (ii) oscillators are identical with respect to their dynamics, and (iii) the function governing the evolution of the internal state of oscillators over time is smooth, monotonically increasing, and concave down.

A study by Bottani followed [41], demonstrating that globally-coupled oscillators with pulse interaction can synchronise under broader conditions than those considered in the theorem of Mirollo and Strogatz. More recently, Lucarelli and Wang [42] showed that local nearest neighbour coupling among oscillators is sufficient, even in systems whose topology changes every T_c time units, provided that the entire network is connected every 2 T_c time units. It should be noted that in all studies described above, for the network of oscillators to converge into a self-synchronised state, one key assumption is that all oscillators have the same or nearly identical frequencies.

12.2.3.2 Definition of Pulse-coupled Oscillators

In a population of pulse-coupled oscillators, each individual i maintains an internal activation level x_i that increases over time until a given threshold is reached. Then, the oscillator i fires, x_i is reset to zero, and the cycle repeats. When a neighbour oscillator j observes the firing of oscillator i, it further increases its own activation

level x_j. If x_j exceeds the firing threshold, the oscillator j fires, resets its activation level to zero, and restarts its behaviour.

Analytically, general pulse-coupled networks can be written as [43]:

$$\dot{x_i} = f_i(x_i) + \epsilon \sum_{j=1}^{N} g_{ij}(x_i)\delta(t - t_j^* - \eta_{ij}) \,, \tag{12.2}$$

where x_i denotes the activation level of oscillator i, and the function f_i describes its dynamics. The *coupling constant* ϵ denotes the strength of interactions between oscillators. N is the set of oscillators neighbours of i. The pulse-coupling function g_{ij} describes the effect of the firing of another oscillator j on i. The time t_j^* marks the moment when oscillator j last fired. When j fires, the activation level x_i is increased by $\epsilon g_{ij}(x_i)$ after a delay $\eta_{ij} \geq 0$. The increment is given in the form of the Dirac delta function δ satisfying $\delta(t) = 0$ for all $t \neq 0$, $\delta(0) = \infty$, and $\int \delta = 1$. In Fig. 12.2, we show an example of the interactions between two oscillators i and j.

12.2.3.3 Robot Model and Individual Behavioural Control

In this study, we model the behaviour of individual robots with pulse-coupled oscillators. Each cell in a given automaton corresponds to one stationary robot. We

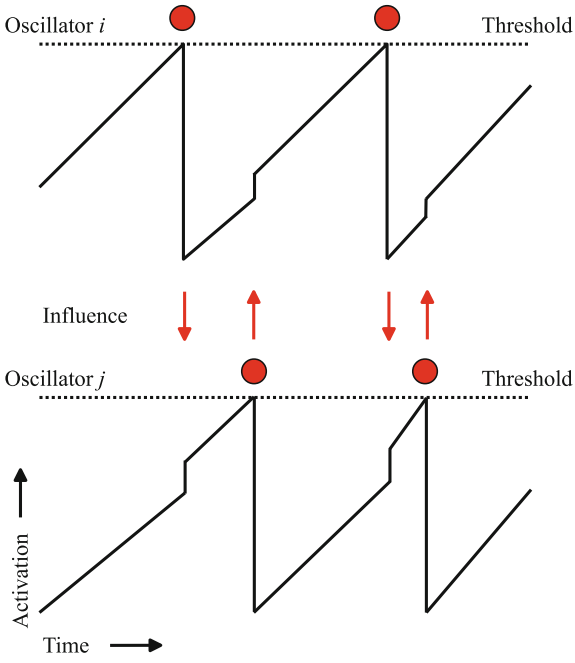


Fig. 12.2 Example of the interactions between two pulse-coupled oscillators i and j. Each oscillator increases its activation level at a constant rate until: (i) the threshold is reached, which resets the activation level to zero, or (ii) until the firing of the other oscillator is detected, at which point the activation of the oscillator that sensed the firing is increased by $\epsilon g(x)$, where ϵ is the coupling constant and $g(x)$ is the pulse-coupling function

model the robots after the combined actuator-sensor units (CASUs), static robotic nodes described by Schmickl et al. [6] in the context of the ASSISIbf project. In our experiments, robots have to collectively adjust and synchronise to produce a local cue, a common vibration pattern. The robotic devices are equipped with: (i) actuators to emit vibrations at a fixed frequency, and (ii) sensors to detect if neighbouring robots are vibrating. Given the discreteness of the cellular automata model, we transform the continuous model for general pulse-coupled systems described in Eq. 12.2 into a discrete time dynamical system with piecewise dynamics, similarly to Christensen et al. [10], as follows:

$$x_i(n+1) = \begin{cases} x_i(n) + \frac{1}{T} + \epsilon\gamma_i(n)g(x_i(n)) & \text{if } x_i(n) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{12.3}$$

where $x_i(n)$ is the activation level of oscillator i at time step n, T is the period between firings of an isolated oscillator, and ϵ is the coupling constant. $\gamma_i(n)$ is the number of neighbours of i that fired in time step n, and $g(x_i(n))$ is the pulse-coupling function. If the activation level of one oscillator exceeds the threshold of 1.0, the corresponding robot vibrates at a fixed frequency, and the state of the cell is set to 1. If the oscillator is not pulsing, the robot is not vibrating and therefore the state of the cell is 0. An example of the development of the activation levels during a run with 100 oscillators is shown in Fig. 12.3. In the example provided, the oscillators synchronise after 1,270 time steps.

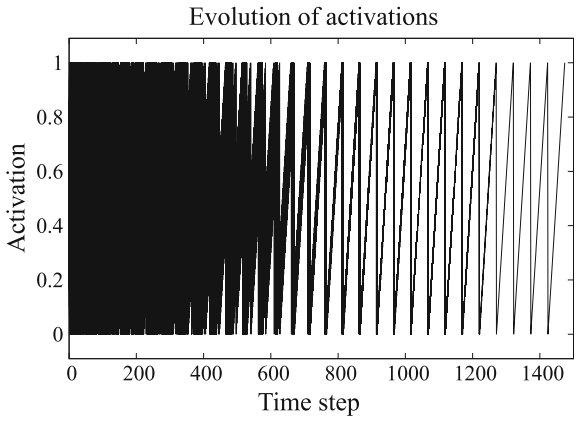


Fig. 12.3 Example of the evolution of activations levels in a population of 100 pulse-coupled oscillators over the course of 1,500 time steps. After time step 1,270, the activation levels overlap, which indicates that the oscillators have synchronised

12.3 Experimental Assessment

In this section, we introduce a number of definitions and notations that we use to analyse the behaviour of cellular automata. In cellular automata, the analytical prediction and classification of behaviour is a complex problem [44]. Depending on the local transition function, aspects such as the update order of the cells, the degree of asynchrony, and the initial configuration can influence the dynamics of the system. In a number of situations, there is even no direct relation between the dynamics of automata following the same local function but subject to different update conditions, see [22, 26, 45] for examples.

12.3.1 Characterising the Behaviour of Cellular Automata

We analyse the behaviour of the automata based both on quantitative and qualitative observations. One of the aspects we are interested in analysing is the time necessary for the system to synchronise under different experimental conditions. We define a cellular automaton of size $N = l \times l$ to exhibit *synchronised* behaviour if: (i) the activation of the first cell is no further than l time steps from all other activations, and (ii) each cell only fires once during this period. Given that each cell only senses the states of its nearest neighbours, our definition of synchronised behaviour takes into account the theoretically *maximum* latency of the system, i.e., the number of time steps necessary for the information to propagate across the automaton.

We estimate the behaviour of different automata and measure the *quality of the synchronisation* process using two methods:

1. We analyse the space-time diagrams of the automata under execution. This form of analysis provides a qualitative impression of behaviour, but does not serve as a formal classification criterion.
2. Complementarily, we quantify the behaviour of the automata using a number of domain-dependent measures, namely:

   - *Convergence rate*, defined as the number of runs in which the automata are able to synchronise their behaviour.
   - *Transient time*, computed as the number of time steps necessary for a given automaton to converge into a synchronised state after it was started.
   - *Speed of collective oscillation*, defined as the total number of time steps necessary for *one instance* of collective synchronised behaviour to be completed, i.e., the time elapsed between the first and the last activation of cells in the automaton.
   - *Period between oscillations*, defined as the number of time steps elapsed *between* different instances of synchronised behaviour. If the automata are properly synchronised, this period should be consistent throughout time, and in accordance with the value T described in Eq. 12.3.

Additionally, cellular automata quantification of behaviour is based on a domain-dependent measure denoted ρ^*. $\rho^* \in [-1, 1]$ measures the degree of correlation between two configurations c_1 and c_2 based on a normalised Hamming distance [46] as follows:

$$\rho^* = 1 - \frac{2 \cdot \sigma_{Ham}(c_1, c_2)}{len(c_1)} , \tag{12.4}$$

$$\sigma_{Ham}(c_1, c_2) = \sum_{i=1}^{len(c_1)} 1 - \delta(c_1[i], c_2[i]) , \tag{12.5}$$

where $\sigma_{Ham}(c_1, c_2)$ is the Hamming distance between c_1 and c_2, $len(c_1) = len(c_2)$ is the length of the configurations, and $\delta(i, j)$ is the Kronecker delta computed as:

$$\delta(i, j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{12.6}$$

$\rho^* = 1$ indicates that c_1 and c_2 have identical values for all cells, and $\rho^* = -1$ means that all cells have complementary states.

ρ^* was used in previous studies in order to: (i) classify the behaviour of one-dimensional cellular automata [47], and (ii) to identify behavioural responses of one-dimensional cellular automata when subject to asynchrony and to noise in the computation of the cell states [22]. In this study, we use ρ^* to quantify cellular automata behaviour in two ways:

1. *Intra-automata correlation*, henceforth intra-ρ^*, defined as the ρ^* value between two configurations of a given automaton at *different* time steps, thereby computing the rate of cells that change or maintain their state at a given time.
2. *Inter-automata correlation*, henceforth inter-ρ^*, computed as the ρ^* value between configurations of two different automata at a given time step. The inter-automata correlation enables the pairwise comparison of the state of two automata and the quantification of the differences between them.

12.4 Effects of the Update Method on Synchronisation of Behaviour

In this section, we analyse the sensitivity of the cellular automata model to different update methods. We investigate: (i) if global synchronisation can arise from local pulse-coupling, (ii) the properties of behaviour under synchronous and asynchronous conditions, and (iii) the implications of using different alternatives to synchronous updating, including to what extent the emergence of synchronisation is affected by the way cells are updated. In addition, we analyse the interplay between different

configurations of parameters in the individual rules of the cells and the characteristics of the global behaviour.

12.4.1 Methods

The experiments are conducted using a fixed lattice with size $N = 10 \times 10$ and toroidal boundary conditions. We conduct experiments using the five different update methods described in Sect. 12.2.2, namely: (i) synchronous, (ii) α-asynchronous, (iii) κ-scaling, (iv) random order asynchronous, and (v) line-by-line sweep. For the α-asynchronous method and the κ-scaling method, the synchrony rate α is varied from $\alpha = 0.99$ (almost synchronous) to $\alpha = 1/N = 0.01$ (approximates sequential, fully asynchronous updating), in decrements of 0.01.

For each configuration in each set of experiments, we conduct 100 independent runs. Each automaton is initialised with a random configuration. A random activation level $\in [0, 1]$ sampled from a uniform distribution is independently assigned to each oscillator at the beginning of each run. The oscillation period T is set to 50 time steps. The coupling constant ϵ is set to 0.1. With respect to the oscillators' dynamics, as defined in Eq. 12.3, we use the linear dynamics given by the pulse-coupling function $g(x) = x$. We experiment with different coupling constants ϵ and distinct pulse-coupling functions in Sect. 12.4.2.2. Each run lasts 50,000 time steps.

12.4.2 Results

Table 12.1 lists the transient time necessary for the automata to synchronise when subject to different update methods. The synchronisation process is, on average, slightly faster if there is a fixed update order, either implicit as in the case of the synchronous update, or explicit as in the line-by-line sweep. Results show that a fixed update order is not always beneficial, as there is convergence in only 92 % of the runs conducted using each method. On the other hand, the random order method displays a convergence rate of 100 %. Differences in the convergence rate of the synchronous and the line-by-line sweep methods are statistically significant

Table 12.1 Transient time and convergence rate of systems under: (i) synchronous update, (ii) line-by-line sweep, and (iii) random order asynchronous update

Update Method	Average	Std. Dev.	Shortest	Longest	Convergence %
Synchronous	985	320	509	2,382	92
Line-by-line	897	292	380	1,793	92
Random Order	1,118	518	378	3,065	100

The values listed are the result of 100 independent runs for each experimental configuration

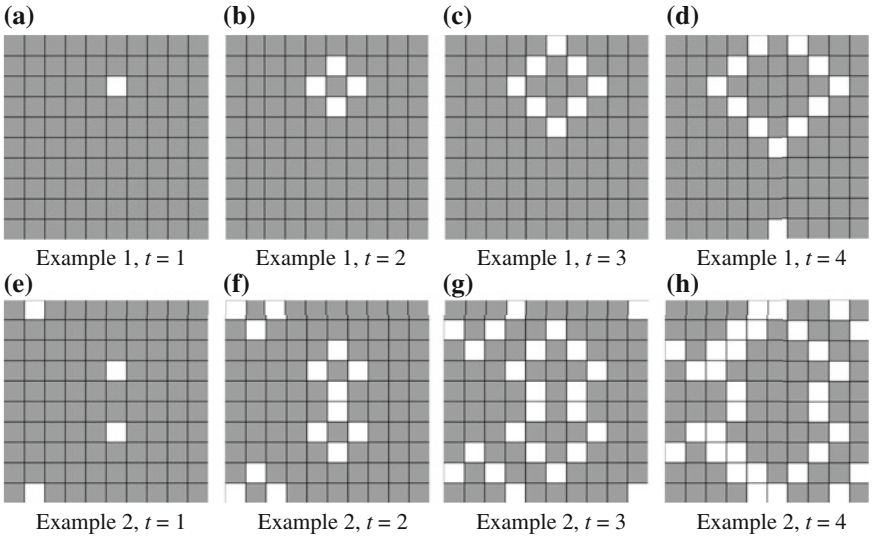


Fig. 12.4 Two examples of the field-based wave-like effect exhibited under synchronous updates. Cells in state 1 (resp. 0) are *coloured* in *white* (resp. *black*), a convention that is maintained throughout the study

with respect to the convergence rate of the random order method (two-tailed Fisher's exact test, $\rho = 6.8 \times 10^{-3}$).

The synchronous and line-by-line sweep methods lead to periodic vibration patterns that repeat every $T = 50$ time steps. Nonetheless, the spatiotemporal patterns exhibited by the automata are due to spurious effects caused by the update method. Two examples are shown in Fig. 12.4 for the synchronous case. The first example describes one particular run in which the automaton evolved into a behaviour with perfectly synchronised neighbour-to-neighbour influence. The first oscillator to fire is the origin of a field-based wave propagation phenomenon (see Sect. 12.2.1 for the definition of a *field*). When the wave starts, all oscillators have activation levels close to the firing threshold. After the firing of the first oscillator, each set of neighbours that have not yet fired will fire in consecutive time steps. The wave propagates throughout the system until all oscillators have fired. In this way, after the initial cell c becomes active at time step t, the behaviour of the automaton at time step $t + i$ will be a field $\varphi(c, i)$, $\forall i \leq 10$. If two or more oscillators fire in the initial time step, as shown in Fig. 12.4e, each of these activations starts a local field-based wave-like phenomenon.

The wave effect is due to the characteristics of the synchronous update method. The state of a given cell at time t can only be perceived by its neighbourhood at time $t + 1$, and therefore the activation of pulse-coupled oscillators is always taken into account with a latency of one time step by its neighbours. As a result, adjacent cells can only become active at consecutive time steps. Depending on how many cells

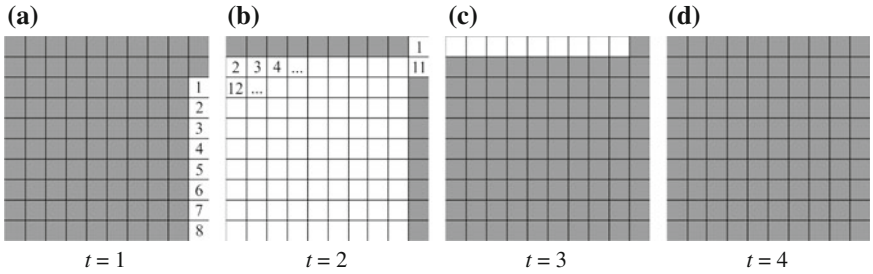


Fig. 12.5 Examples of the domino-like effect caused by the line-by-line sweep. In the first two time steps, $t = 1$ and $t = 2$, the order by which the cells become active in the corresponding time step is marked numerically. Cells in state 1 (resp. 0) are *coloured* in *white* (resp. *black*)

initiate the collective oscillation process and on how many neighbours each cell can cause to activate, the speed of collective oscillation after synchronisation may require from 6 time steps (Fig. 12.4, example 2) to 11 time steps (Fig. 12.4, example 1).

By analysing the sequence of activation of cells at a microscopic level, we observe that, under a line-by-line sweep, there is also a spurious correlation between the update order and the spatiotemporal patterns of the automata. The fixed sequential update order always causes a domino-like effect in the activations of the cells. Figure 12.5 exemplifies the domino-like effect during a collective oscillation behaviour. The order by which the cells become active in the first two time steps is marked numerically (Fig. 12.5a, b). Once a cell c is updated and becomes active, each neighbour of c will also become active when it is updated. In other words, the *directionality* of the influence between cells is solely due to the update order. Depending on the spatial position of the first cell to become active, the speed of collective oscillation varies from $N = 1$ time steps, if the cell is in the top-left corner of the automaton, to $N = 3$ time steps, which is the most frequent behaviour.

Increasing stochasticity in the update process through a continuously randomised order slightly delays convergence by augmenting the transient time. As listed in Table 12.1, random order asynchronous update yields an average transient time of 1,118 time steps. However, unlike the previously analysed update methods, the random order asynchronous scheme does not introduce any artefactual correlation in the dynamics of the cellular automata. Firstly, the automata always converge. Secondly, analysis of the spatiotemporal evolution of the automata shows that every 50 time steps, *all* cells become active in one time step, and therefore there is always a speed of collective oscillation $s = 1$. After the transient time $t_{transient}$ has elapsed, the asymptotic behaviour of synchronised automata is equal, which is detected by the stabilisation of the inter-ρ^* value $=1$ when comparing configurations at each time step $t_{transient} + i$, with $t_{transient} + i \leq 50{,}000$.

The collective oscillation under the random order method is illustrated in Fig. 12.6 through the intra-ρ^* values of one run. The transient time is of approximately 800 time steps. Variations of the intra-ρ^* value until time step 500 indicate that cells become active with no particular order or coupling. Afterwards, oscillators minimise phase

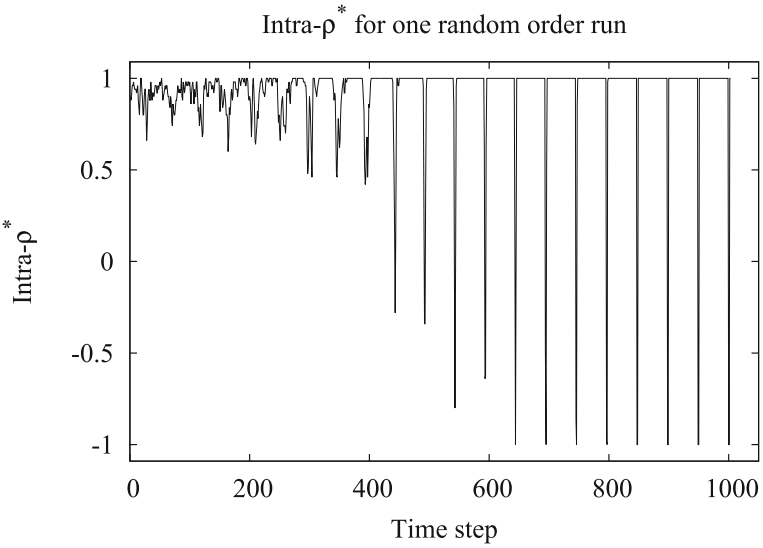


Fig. 12.6 Example of intra-CA correlation over the course of 1,000 time steps under the random order update method. The automata is synchronised after time step 800. Each peak of the intra-ρ^* value from 1 to -1 and vice-versa that occurs every 50 time steps represents one instance of collective oscillation (see text)

differences, and cells start to exhibit a progressively more synchronised behaviour. At time step 800, cells are completely synchronised based on an asynchronous process and local interactions. Each peak of the intra-ρ^* value from 1 to -1 and vice-versa represents one instance of collective oscillation. At this point, all cells are in state 0, i.e., no cell is active. In the following time step, all cells become active and therefore intra-ρ^* value $= -1$. A value of -1 is also measured in the subsequent time step because all cells move from an active to an inactive state, after which the intra-ρ^* value $=1$ is maintained until the next collective activation. The behaviour is then repeated every 50 time steps until the end of the simulation.

At a microscopic scale, during the activation time step the order by which the cells become active is determined by the randomly generated update sequences. These results support the work of Cornforth et al. [13] and Harvey and Bossomaier [48], among others, which have suggested that random order methods can generate quasi-cyclic behaviour. Under the light of the random order method, the automata can be said to display a truly emergent synchronised behaviour. One positive effect of this update method is that, as mentioned above, it removes the spurious effects that could be linked to a particular update order. Secondly, the random order scheme appears to be more faithful to the physical reality of the system being modelled than both the synchronous and the line-by-line methods. As initially argued by Kanada [12], the fact that the major source of randomness lies in the order of state transitions of the cells allows to take into account the microscopic structure of systems, and is therefore suitable for modelling purposes. However, it should be noted that in a

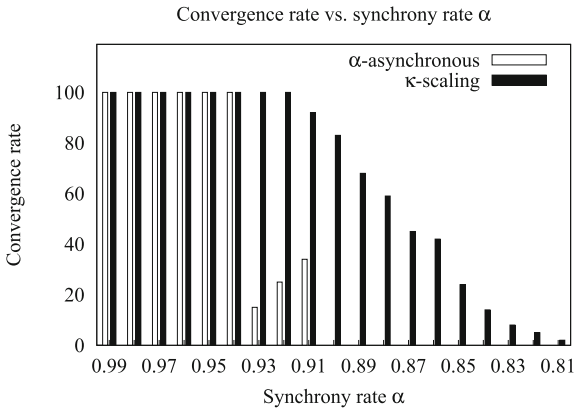


Fig. 12.7 Convergence rate of α-asynchronous and κ-scaling systems when subject to distinct degrees of asynchrony. Convergence cannot be ensured for α-asynchronous systems if $\alpha \leq 0.93$, and for κ-scaling systems if $\alpha \leq 0.91$

real distributed system, it may be difficult to ensure a completely random *sequential* updating order.

12.4.2.1 α-dependency

In this section, we analyse the behavioural response of the cellular automata when subject to α-asynchrony-based update methods. Results show that under the α-asynchronous and the κ-scaling methods, the behaviour of the automata significantly depends on the degree of asynchrony in their update. With increasing α-asynchrony, the automata evolve a behaviour noticeably different from that produced by the three update methods analysed above. As shown in Fig. 12.7, for $0.90 < \alpha \leq 0.93$, the convergence rate of α-asynchronous systems varies from 15 % to 34 %, which is significantly lower than the convergence rate of automata subject to higher α values (two-tailed Fisher's exact test, $\rho < 1.0 \text{ x } 10^{-4}$ for all $0.90 < \alpha \leq 0.93$). In effect, for $\alpha \leq 0.90$, α-asynchronous systems can no longer synchronise in the 50,000 time steps limit. Performing a temporal compensation and corresponding normalisation of the sampling differences by means of the κ-scaling method counteracts to some extent the effects of α-asynchrony. The automata can always overcome asynchrony for $\alpha > 0.91$. As the synchrony rate is lowered, the convergence rate of κ-scaling systems decreases proportionally to α.

From a practical point of view, α-asynchronous and κ-scaling methods show how sensitive cellular automata can be to asynchrony. Table 12.2 lists the average transient time and the average period for $0.99 \leq \alpha \leq 0.94$. Aside from the aforementioned convergence rate, asynchrony by means of less cell activity significantly increases both the transient time and the average period length between consecutive group-level activations. Even for minimal asynchrony, i.e., $\alpha = 0.99$, the average period is 128 time steps for α-asynchronous systems and 124 time steps for κ-scaling systems. With increasing asynchrony, the automata become increasingly lagged thus resulting

Table 12.2 Transient time and length of the period between consecutive group-level oscillations

Synchrony rate α	α-asynchronous		κ-scaling	
	Transient time	Period	Transient time	Period
0.99	1,090	128	1,152	124
0.98	1,437	345	1,340	302
0.97	1,989	863	1,855	680
0.96	3,495	2,294	2,404	1,471
0.95	12,189	6,519	3,876	2,755
0.94	19,278	9,753	4,931	4,387

Values are the average of 100 independent runs for each experimental configuration

in significantly higher transient times and periods between collective oscillations. For $\alpha = 0.94$, α-asynchronous systems have a transient time of approximately 19,278 time steps and a period of 9,753 time steps. κ-scaling systems are less affected as they display a transient time of 4,931 time steps and a period of 4,387 time steps.

Our analysis shows that: (i) the assessed α-asynchrony-based update methods do not enable the systems to evolve towards a global vibration pattern, and (ii) increasing asynchrony further degrades behaviour. In fact, the existence of contrasting behaviours under α-asynchronous dynamics, even when using moderate asynchrony values, indicates a hidden sensitivity that may need to be handled carefully depending on the context.

Overall, the results presented in this section corroborate that not only is asynchrony relevant, but that the *type* of asynchrony can have significant effects when modelling multiagent or multirobot systems by means of discrete dynamical systems. This question takes a special importance due to the fact that, as argued by Cornforth et al. [13], multiagent systems are usually modelled based on the synchronous update method. This begs the question: when modelling multiagent systems that have to achieve synchronisation or consensus, in which cases is the multiagent system robust to modifications in the updating? Complementarily, in such multiagent systems, to what extent is the behaviour exhibited under asynchronous dynamics dependent on the properties of the local transition function? The following section addresses the second question by analysing the robustness and the global dynamics of the system when subject to variations in the main parameters of the pulse-coupled oscillators: the coupling constant ϵ and the coupling function $g(x)$.

12.4.2.2 Assessing the Dynamics of the Random Order Method

The behaviour of the oscillators as described in Eq. 12.3 depends on the interactions between neighbouring cells. The strength of each interaction is controlled by the coupling constant ϵ and by the coupling function $g(x)$. In this section, we assess in which conditions the variation of ϵ and of $g(x)$ alters the dynamics of the automata.

Table 12.3 Convergence rate of different pulse-coupling dynamics as a function of the coupling constant ϵ

Coupling constant ϵ	Convergence rate (%)		
	Linear function	Sigmoid function	Sine function
0.01	14	0	0
0.02	99	33	0
0.05	100	100	0
0.1	100	100	0
0.2	100	100	0
0.5	100	100	100
1.0	100	100	100

For each experimental configuration, the convergence rate is measured by performing 100 independent runs

Our goal is to examine the robustness or sensitivity of the update method to changes in the rules that govern the behaviour of cells.

We conduct the experiments using the update method that yielded the most reliable result, namely the random order asynchronous method. The effects of the coupling constant are assessed for $\epsilon \in \{0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.0\}$. We also study three pulse-coupling functions. The choice of the coupling function was made to test different dynamics, namely: (i) linear dynamics with $g(x) = x$, (ii) even symmetrical dynamics with respect to $x = 0.5$ based on a sine function $g(x) = \sin(\pi x)$, and (iii) "S-shaped" dynamics given by the sigmoid function $g(x) = \frac{1}{1+e^{(6-12x)}}$. The three functions are shown in Fig. 12.8. For each experimental configuration, we perform 100 independent runs. Oscillators are initialised as defined in Sect. 12.4.1, i.e., with random initial activation levels, and with the period T set to 50 time steps. Each run lasts 50,000 time steps.

Table 12.3 summarises the convergence rate when ϵ and $g(x)$ are varied. The linear pulse-coupling is the most robust function as it shows the highest overall convergence rate in our experiments. Except for the lowest value $\epsilon = 0.01$, convergence is almost always ensured. The sigmoid function and the sine function ensure con-

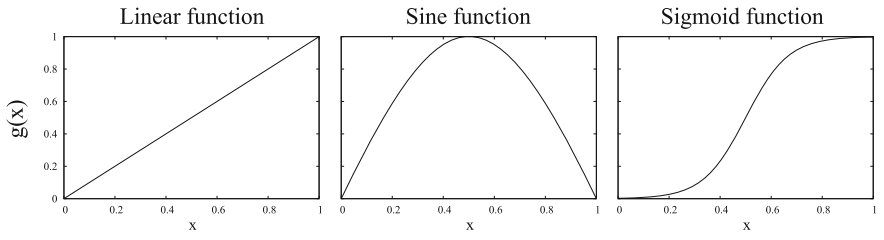


Fig. 12.8 The three models of pulse-coupling functions assessed. From *left* to *right*: linear dynamics, even symmetrical dynamics with respect to $x = 0.5$ based on a sine function $g(x) = \sin(\pi x)$, and "S-shaped" dynamics given by the sigmoid function $g(x) = \frac{1}{1+e^{(6-12x)}}$

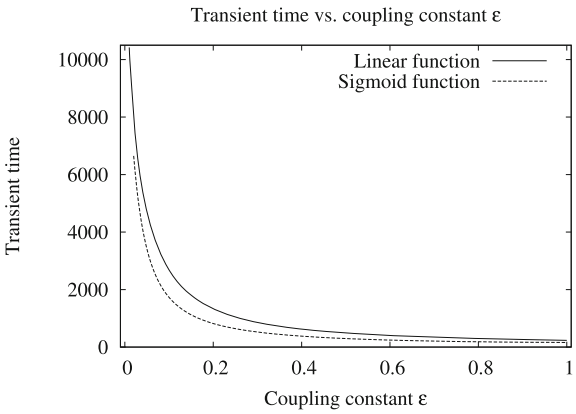


Fig. 12.9 Average transient time as a function of the coupling constant ϵ. For the linear function, the average transient time varies from 10,423 time steps for $\epsilon = 0.01$ to 234 time steps for $\epsilon = 1$. For the sigmoid function, the average transient time is of 6,651 time steps for $\epsilon = 0.02$ and of 161 time steps for $\epsilon = 1.0$. For the sine function (not plotted), the average transient time varies from 326 time steps ($\epsilon = 0.5$) to 96 time steps ($\epsilon = 1$)

vergence if $\epsilon \geq 0.05$ and $\epsilon \geq 0.5$, respectively. With respect to the time necessary for convergence, Fig. 12.9 shows the average transient time for the linear function and for the logistic function when using different coupling values. For these two functions, the lowest transient time is given when the oscillators are strongly coupled. For $\epsilon = 1$, the average transient time is 234 time steps for the linear function and 161 time steps for the sigmoid function. The results suggest that high values of ϵ are preferable in the synthesis of synchronised behaviour. For the two functions, the transient time decreases exponentially with the increase of the coupling constant until $\epsilon = 0.6$, after which point the values vary significantly less.

Despite the substantial reduction of the transient time due to the increase of the coupling constant when using either a linear function or a sigmoid function, the fastest overall convergence is given by the even symmetrical dynamics of the sine function. The average transient time varies from 326 time steps for $\epsilon = 0.5$ to 96 time steps for $\epsilon = 1$. The reduction in transient time is due to the characteristics of the sine function, which enable a more effective adjustment of the activation level. The line of symmetry dictates that increasingly higher adjustments are made to the activation level for $0 < x < 0.5$. For $x > 0.5$, the closer the activation level x is of the threshold of 1, the more subtle the adjustment is.

The analysis conducted in this section shows that the parameters ϵ and $g(x)$ can have a profound effect on the convergence rate and transient time of the automata. We estimated and analysed the spectrum of convergence rates and transient times. Firstly, the results show that increasing the strength of the interaction between neighbouring cells is beneficial as if ϵ is large, the oscillators tend to synchronise faster. Secondly, coupling functions with different properties enable slower or faster convergence. Thirdly, it should also be noted how low coupling values affect the behaviour exhib-

ited by an automaton. For $\epsilon = 0.01$ and $\epsilon = 0.02$, the speed of collective oscillation varies from 1 to 11 time steps for the linear function and from 1 to 7 time steps for the sigmoid function. In other words, low coupling constants cause instabilities in the systems as the automata evolve towards different configurations. Consequently, the periodic, quasi-cyclic patterns and the repetition of series of configurations are altered by using different parameters.

12.4.3 Summary

In this section, we examined the effects of the update method in the synthesis of synchronised behaviour. We showed that in the modelling of multirobot or multiagent systems, not only is asynchrony relevant, but that the type of asynchrony has profound effects on dynamics. These results are especially important if we consider that multiagent systems are usually modelled using synchronous update methods [13].

In our experimental setup, the random order asynchronous method was shown to be the most reliable method. We showed that both the synchronous and the line-by-line schemes introduce artificial structure in the evolution of our cellular automata model. We also showed that, in our modelling scenario, α-asynchronous and κ-scaling methods can also have significant effects on the behaviour of the system. The two methods exhibited a hidden sensitivity, even for high asynchrony values, that should be handled carefully when modelling systems that are continuous in time and space.

To conclude the section, we experimentally analysed the importance of the coupling constant and of the pulse-coupling function. Chiefly, our results showed that: (i) stronger interactions between neighbours are preferable, as they significantly increase both the convergence rate and the transient time, and (ii) that the type of dynamics of the coupling function play a key role in the synchronisation process.

12.5 Perturbing the Topology

In this section, we investigate how topology perturbations modify the evolution of synchronised behaviour in our cellular automata model. Our goal is to determine what happens when the cellular automata are no longer perfectly synchronous and the topology is perturbed, i.e., to estimate the stability or sensitivity of the systems when topology characteristics become irregular.

12.5.1 Methods

We perturb the topology by definitively removing connections between cells. Let $G = (V, E)$ be the oriented graph that represents the connections between

cells. For all cells $c_i, c_j \in V$, with $c_i \neq c_j$, the connection $(c_i, c_j) \in E$ if and only if c_j belongs to the neighbourhood of c_i. The oriented graph with perturbed topology $G_{pert} = (V, E_{pert})$ is obtained by assigning to each cell $c_i \in V$ an independent probability of removing a randomly chosen connection between c_j and one of its neighbours c_j. The parameter p_{cr} is defined as the *connection removal* rate. It should be noted that the discrete pulse-coupled model defined in Eq. 12.3 is a totalistic rule in the sense that it takes into account the total of active neighbours. A neighbour that is not sensed due to a connection removed is considered as being always in state 0. In this way, perturbing the topology of the automata is conceptually similar to simulating faults in the sensors of the robots, one of the central issues with autonomous robots [49].

We conduct two series of experiments, henceforth *dynamic* perturbations setup and *fixed* perturbations setup. In the dynamic perturbations setup, there is a continual removal of connections at *each* time step. Our objective is to investigate the behaviour of the cellular automata when topology characteristics are continuously modified throughout time. It should be noted that in this experimental setup, the connections graph may become disconnected as a result of the deletion of connections between cells. Therefore, the major requirement in the task is for each cell to synchronise with its neighbours before the neighbours stop being sensed due to the lack of connections. On the other hand, in the fixed perturbations setup we analyse the convergence properties of the systems when the topologies are made irregular *but* do not change during the course of a simulation. In this setup, connections are probabilistically removed according to the connection removal rate p_{cr} before the simulation starts.

The experimental protocol is here defined using a lattice with size $N = 10 \times 10$ and toroidal boundary conditions. Experiments are conducted using the random order asynchronous method and the linear pulse-coupling function $g(x) = x$, as this was shown to be the most robust configuration. In the dynamic perturbations setup, the connection removal rate p_{cr} is varied in $[0.0, 0.1]$ in steps of 0.002. As the convergence properties of the system is dependent on the strength of the interactions between neighbouring cells, we also vary the coupling constant $\epsilon \in [0.1, 1.0]$ in steps of 0.1. In the fixed perturbations setup, the connection removal rate is larger because connections are only removed once, i.e., before the simulation starts. The connection removal rate is varied in $[0.0, 0.75]$ in steps of 0.05. We experimentally verified that for $p_{cr} > 0.75$, the graph tends to be disconnected, which we do not allow because convergence would not be possible. For each configuration in each set of experiments, we conduct 100 independent runs. The parameters and initialisation of the automata follow the experimental setup described in Sect. 12.4.1. Each run lasts 50,000 time steps.

12.5.2 Results

We start by analysing the results of the dynamic perturbations setup. In Fig. 12.10, we separately represent the convergence rate and the transient time as a function of

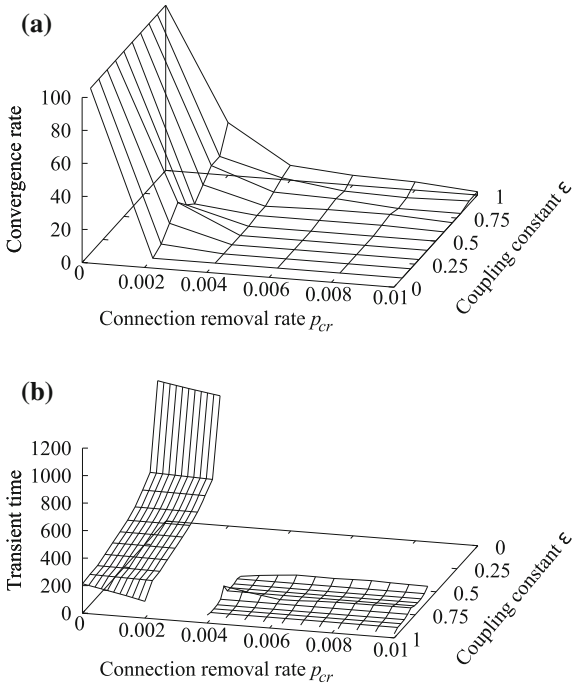


Fig. 12.10 Dynamic perturbations setup. Sampling surface of: (*a*) the convergence rate sampling surface for the dynamic perturbations setup, and (*b*) the average transient time sampling surface for the dynamic perturbations setup. The values displayed in each sampling surface are a function of both the coupling constant ϵ and the connection removal rate p_{cr}. **a** for $p_{cr} > 0.01$ the sampling surface is flat, which indicates 0 % of convergence—not shown for better visualisation of the sampling surface. **b** the sampling surface shows a discontinuity of behaviour for $0.002 < p_{cr} < 0.004$, which is marked by separating the sampling surface into two parts. Note that the range of the y-axis is reversed, i.e., values regarding the coupling constant ϵ are plotted from $y = 1$ to $y = 0$

the connection removal rate p_{cr} and of the coupling constant ϵ. Each set of values obtained is represented in a three-dimensional space, which is projected on a two-dimensional *sampling surface*.

Results show that the cellular automata are sensitive to perturbations in the topology. In general, higher coupling values ϵ enable the automata to synchronise more often. However, even for the smaller value of $p_{cr} = 0.002$ considered, the maximum convergence rate is of 32 % (for $\epsilon = 1.0$). For $p_{cr} = 0.004$, there is no synchronisation of behaviour for $\epsilon \leq 0.5$ and the highest convergence rate is of 9 % ($\epsilon = 1.0$). For higher connection removal rates, the convergence rate continuously decreases. For $p_{cr} > 0.01$, there is no convergence and the sampling surface is flat and horizontal.

The impact of removing connections is two-fold. An interesting effect of perturbing the topology is that, as shown in Fig. 12.10b, increasing rates of connection removal can harm the overall convergence rate *but* reduce the transient time when the automata do converge. Visual examination of the transient time sampling surface

shows a discontinuity of behaviour, which is marked by separating the surface into two parts. If no connections are removed, the minimum average transient time is 213 time steps ($\epsilon = 1$). For $p_{cr} = 0.002$, the transient time evolves in a similar manner except that the minimum average transient time is of 122 time steps, again for $\epsilon = 1.0$. For the critical values of $p_{cr} \geq 0.004$, for which there is no convergence for $\epsilon \leq 0.5$, the transient time is consistently low and decreases to an average of 30 time steps for the case $p_{cr} = 0.01$, $\epsilon = 1.0$.

The results obtained in the dynamic perturbations setup are the consequence of multiple intricate factors related to local synchronisation of cells. Removing connections can either deteriorate or accelerate convergence depending on the context of the cells. For example, consider a cell c_i and two of its neighbours, c_j and c_k. Cells c_i and c_j have a small lag between their activations, i.e., they are not entirely synchronised, and c_k is significantly delayed with respect to the firings of c_i and c_j. If the connection (c_i, c_j) is removed, c_i will have to synchronise its behaviour *only* with c_k, which will require an amount of time that depends on the phase differences between the two cells. During this time, depending on the connection removal rate, the connection (c_i, c_k) may be removed before the cells are synchronised, causing them to remain unsynchronised throughout time. As a result, there will be no convergence in the automaton.

If the first connection removed refers to two neighbours that have a large phase difference, namely c_i and c_k, synchronisation is accelerated because c_i synchronises with c_j without the "interference" of c_k. If the automata at a global level are subject to more acceleration than deterioration phenomena, then a self-organised behaviour will emerge faster as the result of the interaction between the cells at a local level. On the other hand, if there are more deterioration than acceleration phenomena in the relation between the cells, convergence is made progressively more difficult because the graph becomes increasingly disrupted and potentially disconnected. Therefore, in the dynamic perturbations setup, the acceleration and deterioration aspects described above are the result of a *continuous* removal of connections. Nonetheless, one important question remains: if the topology is perturbed but afterwards remain unchanged, under which conditions can the cells synchronise their behaviour?

In the fixed perturbations setup, the automata *always* converge regardless of the experimental configuration, which results in a completely horizontal sampling surface. For the most extreme case assessed, $p_{cr} = 0.75$, on average 75 out of 400 connections are removed from the initial topology, equivalent to 18.75 %. This aspect is indicative of the robustness of the cellular automata model and of its ability converge, as long as the connection graph is connected.

Figure 12.11 shows the sampling surface describing the evolution of the transient time. In the fixed perturbations setup, contrary to the dynamic perturbations experiments, the transient time tends to increase if higher ϵ and p_{cr} values are used. This increase is more accentuated for coupling constants around $\epsilon \approx 0.75$. By analysing the speed of collective oscillation and the period between consecutive oscillations, we observe that for $p_{cr} > 0.4$ and $\epsilon > 0.75$, the system becomes unstable. For these values, the average period varies from 100 to 180 time steps, and collective oscillation can require up to 10 time steps. The results show the interplay between the

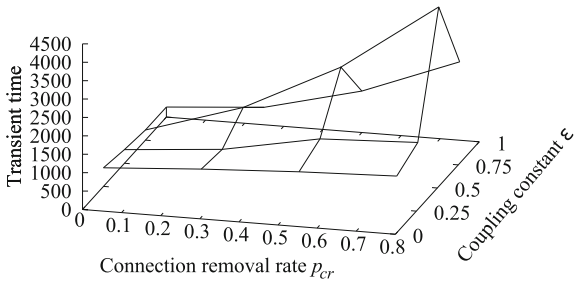


Fig. 12.11 Fixed perturbations setup. Sampling surface of the average transient time. The values displayed in the sampling surface are a function of both the coupling constant ϵ and of the connection removal rate p_{cr}

coupling constant and the *degree* of connectivity of the connections graph, thereby indicating that the topology indeed plays a central role in the synchronisation properties of the cellular automata. In automata with regular topologies such as those used in Sect. 12.4, higher coupling constants ϵ are beneficial and accelerate convergence. However, if the connectivity of the automata is irregular, then setting ϵ too high is problematic because each cell has a significant effect on its neighbours. The high degree of influence in the behaviour of neighbouring cells leads to phase instability, and can drive the system towards either acceleration of convergence or deterioration of behaviour.

12.5.3 Summary

In this section, we analysed the effects of perturbing the cellular automata topology by probabilistically removing connections between neighbouring cells. Based on extensive numerical simulations, we showed that topology characteristics are important for the emergence of synchronised behaviour. If the irregularity of the topology increases over time, the impact is often two-fold. The convergence rate decreases with the increase of irregularity but the transient time is effectively smaller when the automata manage to converge. On the other hand, if the degree of irregularity is kept fixed, convergence is ensured as long as the connections graph does not become disconnected. Additionally, results showed that the strength of the interaction between neighbouring cells by means of the coupling constant ϵ plays a central role in the two circumstances. In particular cases, high values of ϵ either accelerate synchronisation of behaviour, or introduce instabilities in the system. These fluctuations significantly increase the time necessary for convergence and the way by which instances of collective behaviour are produced. Therefore, depending on the topology of the cellular automata, it is necessary to compromise between the strength of local interactions and the degree of connectivity of the network.

12.6 Discussion

A long-standing question in the field of complex systems is determining whether cellular automata are appropriate modelling tools for multiagent systems or if they are not sufficiently expressive. To answer the question, it is first necessary to carefully assess the robustness of cellular automata in a modelling context, and to cover a broad number of conditions in order to identify the *limits* of the modelling tool.

This chapter is a contribution towards the understanding of the effects of the update method and of the topological structure when modelling real-world complex systems with cellular automata. We analysed the collective dynamics of a group of stationary robots inspired by studies in mixed natural-artificial societies [6]. The system was composed by 100 robots, and the behaviour of individual robots was modelled as a pulse-coupled oscillator. We addressed the problem of self-organised synchronisation, in which robots had to adjust their behaviour to produce a population-wide common vibration pattern based on local interactions. We focused on two fundamental aspects: (i) the effects of different update methods, including the interplay between parameters of local rules and the global behaviour, and (ii) the transition from regular to irregular grids by means of dynamic perturbations and fixed perturbations.

The first set of experiments outlined in this chapter demonstrated the impact of five different update methods. Results showed that the way robots synchronise their behaviour can be bounded by the singular properties of the update method. Modifying the state of the cells according to a random update method was shown to be the most robust approach with respect to producing the desired dynamics in the multirobot system. In the second part of the chapter, we conducted a systematic study on the robustness of cellular automata to topology perturbations under a number of different conditions. We concluded that the topology characteristics have various effects on the behaviour of the automata. When subject to topology perturbations, behaviour exhibited may vary from extremely sensitive to extremely robust. Therefore, although few researchers have studied this aspect [29], the topology structure is deemed essential for future studies in modelling the behaviour of multiagent systems with cellular automata.

The broader agenda for future study on more realistic modelling of real-world complex systems with cellular automata is to gain new insights on the emergent behaviour of large-scale multirobot systems, both stationary and mobile, by analysing them from a dynamical systems perspective. In this respect, the topology perturbation experiments can be extended to simulate scenarios in which a failed robot is repaired or replaced by a new one, or the fault is transient, and therefore the cellular automata recover part of their previously lost connections. We also intend to address increasingly more complex tasks in which robots have to achieve synchronisation or consensus. Instead of manually tuning the parameters of individual behaviour, our goal is to synthesise by means of machine learning techniques, the specific parameters of the pulse-coupled oscillators (or other models) that can generate a set of target spatiotemporal dynamics. One important part of our future work is to analyse

the generality and robustness of the learned parameters by assessing them under different update methods and topological structures.

Acknowledgments This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) under the grants SFRH/BD/89573/2012, PEst-OE/EEI/UI0434/2011, PEst-OE/EEI/LA0008/2013, and EXPL/EEI-AUT/0329/2013, and by the European Union —Information and Communication Technologies project 'ASSISIbf', no. 601074.

References

1. Fatès, N., Chevrier, V.: How important are updating schemes in multi-agent systems? An illustration on a multi-turmite model. In: 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 533–540. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2010)
2. Spicher, A., Fatès, N., Simonin, O.: Translating discrete multi-agents systems into cellular automata: application to diffusion-limited aggregation. In: Filipe, J., Fred, A., Sharp, B. (eds.) Agents and Artificial Intelligence. Communications in Computer and Information Science, vol. 67, pp. 270–282. Springer, Berlin, Germany (2010)
3. Tosic, P.: On modeling large-scale multi-agent systems with parallel, sequential and genuinely asynchronous cellular automata. Acta Phys. Pol. B Proc. Suppl. **4**(2), 217–235 (2011)
4. Bouré, O., Fatès, N., Chevrier, V.: First steps on asynchronous lattice-gas models with an application to a swarming rule. Nat. Comput. **12**(4), 551–560 (2013)
5. Chevrier, V., Fatès, N.: Multi-agent systems as discrete dynamical systems: Influences and reactions as a modelling principle. Technical report, INRIA-LORIA (2008)
6. Schmickl, T., Bogdan, S., Correia, L., Kernbach, S., Mondada, F., Bodi, M., Gribovskiy, A., Hahshold, S., Miklic, D., Szopek, M., Thenius, R., Halloy, J.: ASSISI: Mixing animals with robots in a hybrid society. In: Second International Conference on Biomimetic and Biohybrid Systems. volume 8064 of Lecture Notes in Computer Science, pp. 441–443. Springer, Berlin, Germany (2013)
7. Pikovsky, A., Rosenblum, M., Kurths, J.: Synchronization: A Universal Concept in Nonlinear Sciences. Cambridge University Press, Cambridge, UK (2003)
8. Tyrrell, A., Auer, G.: Imposing a reference timing onto firefly synchronization in wireless networks. In: 65th IEEE Vehicular Technology Conference, pp. 222–226. IEEE Press, Piscataway, NJ (2007)
9. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-inspired sensor network synchronicity with realistic radio effects. In: 3rd International Conference on Embedded Networked Sensor Systems, pp. 142–153. ACM Press, New York, NY (2005)
10. Christensen, A.L., O'Grady, R., Dorigo, M.: From fireflies to fault-tolerant swarms of robots. IEEE Trans. Evol. Comput. **13**(4), 754–766 (2009)
11. Mirollo, R., Strogatz, S.: Synchronization of pulse-coupled biological oscillators. SIAM J. Appl. Math. **50**(6), 1645–1662 (1990)
12. Kanada, Y.: The effects of randomness in asynchronous 1D cellular automata. Technical report, Tsukuba Research Center (1997)
13. Cornforth, D., Green, D.G., Newth, D.: Ordered asynchronous processes in multi-agent systems. Phys. D Nonlinear Phenom. **204**(1), 70–82 (2005)
14. Correia, L.: Self-organised systems: fundamental properties. Revista de Ciências da Computação **1**(1), 9–26 (2006)
15. Fatès, N.: A guided tour of asynchronous cellular automata. In: 19th International Workshop on Cellular Automata and Discrete Complex Systems. volume 8155 of Lecture Notes in Computer Science, pp. 15–30. Springer, Berlin, Germany (2013)

16. Bandini, S., Bonomi, A., Vizzari, G.: An analysis of different types and effects of asynchronicity in cellular automata update schemes. Nat. Comput. **11**(2), 277–287 (2012)
17. Dennunzio, A., Formenti, E., Manzoni, L.: Computing issues of asynchronous CA. Fundam. Informaticae **120**(2), 165–180 (2012)
18. Dennunzio, A., Formenti, E., Manzoni, L., Mauri, G.: m-asynchronous cellular automata: from fairness to quasi-fairness. Nat. Comput. **12**(4), 561–572 (2013)
19. Fatès, N., Morvan, M.: An experimental study of robustness to asynchronism for elementary cellular automata. Complex Syst. **16**(1), 1–27 (2005)
20. Ingerson, T.E., Buvel, R.L.: Structure in asynchronous cellular automata. Phys. D Nonlinear Phenom. **10**(1–2), 59–68 (1984)
21. Schönfisch, B., de Roos, A.: Synchronous and asynchronous updating in cellular automata. Biosystems **51**(3), 123–143 (1999)
22. Silva, F., Correia, L.: An experimental study of noise and asynchrony in elementary cellular automata with sampling compensation. Nat. Comput. **12**(4), 573–588 (2013)
23. Bersini, H., Detours, V.: Asynchrony induces stability in cellular automata based models. In: 4th International Conference on Simulation and Synthesis of Living Systems, pp. 382–387. MIT Press, Cambridge, MA, (1994)
24. Ruxton, G., Saravia, L.: The need for biological realism in the updating of cellular automata models. Ecol. Modell. **107**(2–3), 105–112 (1998)
25. Fatès, N.: Directed percolation phenomena in asynchronous elementary cellular automata. In: 7th International Conference on Cellular Automata for Research and Industry. volume 4173 of Lecture Notes in Computer Science, pp. 667–675. Springer, Berlin, Germany (2006)
26. Fatès, N.: Asynchrony induces second order phase transitions in elementary cellular automata. J. Cell. Automata **4**(1), 21–38 (2009)
27. Fatès, N., Regnault, D., Schabanel, N., Thierry, E.: Asynchronous behavior of double-quiescent elementary cellular automata. In: 7th Latin American Symposium. volume 3887 of Lecture Notes in Computer Science, pp. 455–466. Springer, Berlin, Germany (2006)
28. Regnault, D.: Proof of a phase transition in probabilistic cellular automata. 17th: International Conference on Developments in Language Theory. volume 7907 of Lecture Notes in Computer Science, pp. 433–444. Springer, Berlin, Germany (2013)
29. Fatès, N.: Critical phenomena in cellular automata: perturbing the update, the transitions, the topology. Acta Phys. Pol. B Proc. Suppl. **3**(2), 315–325 (2010)
30. Fatès, N.: Does life resist asynchrony? In: Adamatzky, A. (ed.) Game of Life Cellular Automata, chapter 14, pp. 257–274. Springer, London, UK (2010)
31. Fatès, N., Gerin, L.: Examples of fast and slow convergence of 2D asynchronous cellular systems. J. Cell. Automata **4**(4), 323–337 (2009)
32. Belgacem, S., Fatès, N.: Robustness of multi-agent models: the example of collaboration between turmites with synchronous and asynchronous updating. Complex Syst. **21**(3), 165–182 (2012)
33. Kitagawa, T.: Cell space approaches in biomathematics. Math. Biosci. **19**(1), 27–71 (1974)
34. Seybold, P., Kier, L., Cheng, C-K.: Simulation of first-order chemical kinetics using cellular automata. J. Chem. Inf. Comput. Sci. **37**(2), 386–391 (1997)
35. Rajewsky, N., Santen, L., Schadschneider, A., Schreckenberg, M.: The asymmetric exclusion process: comparison of update procedures. J. Stat. Phys. **92**(1–2), 151–194 (1998)
36. Strogatz, S., Stewart, I.: Coupled oscillators and biological synchronization. Sci. Am. **269**(6), 102–109 (1993)
37. Smith, H.M.: Synchronous flashing of fireflies. Science **82**(2120), 151–151 (1935)
38. Glass, L.: Synchronization and rhythmic processes in physiology. Nature **410**(6825), 277–284 (2001)
39. Strogatz, S.: From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. Phys. D Nonlinear Phenom. **143**(1–4), 1–20 (2000)
40. Peskin, C.S.: Mathematical Aspects of Heart Physiology. Courant Institute Lecture Notes. Courant Institute of Mathematical Sciences, New York, NY (1975)

41. Bottani, S.: Pulse-coupled relaxation oscillators: from biological synchronization to self-organized criticality. Phys. Rev. Lett. **74**(21), 4189–4192 (1995)
42. Lucarelli, D., Wang, I-J.: Decentralized synchronization protocols with nearest neighbor communication. In: 2nd International Conference on Embedded Networked Sensor Systems, pp. 62–68. ACM Press, New York, NY, (2004)
43. Izhikevich, E.: Weakly pulse-coupled oscillators, FM interactions, synchronization, and oscillatory associative memory. IEEE Trans. Neural Networks **10**(3), 508–526 (1999)
44. Fuks, H., Skelton, A.: Orbits of the Bernoulli measure in single-transition asynchronous cellular automata. In: 17th International Workshop on Cellular Automata and Discrete Complex Systems, Discrete Mathematics and Theoretical Computer Science, pp. 95–112, (2011). Available from: http://www.dmtcs.org/dmtcs-ojs/index.php/proceedings/article/view/dmAP0107
45. Bouré, O., Fatès, N., Chevrier, V.: Probing robustness of cellular automata through variations of asynchronous updating. Nat. Comput. **11**(4), 553–564 (2012)
46. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)
47. Silva, F., Correia, L.: A study of stochastic noise and asynchronism in elementary cellular automata. In: 10th International Conference on Cellular Automata for Research and Industry. volume 7495 of Lecture Notes in Computer Science, pp. 679–688. Springer, Berlin, Germany (2012)
48. Harvey, I., Bossomaier, T.: Time out of joint: attractors in asynchronous random boolean networks. In: 4th European Conference on Artificial Life, pp. 67–75. MIT Press, Cambridge, MA, (1997)
49. Carlson, J., Murphy, R.R., Nelson, A.: Follow-up analysis of mobile robot failures. In: IEEE International Conference on Robotics and Automation, pp. 4987–4994. IEEE Computer Society Press, Los Alamitos, CA, (2004)

Chapter 13
Cellular Automaton Manipulator Array

Ioannis Georgilas, Andrew Adamatzky and Chris Melhuish

Abstract We present a cellular automaton architecture for massive-parallel manipulation tasks. The cellular-automaton manipulator is an array of actuators, which interact locally with each other and generate coordinated manipulation forces for precise translation of the manipulated object. The cellular-automaton actuator arrays behave as an excitable medium, where initial perturbation leads to propagation of excitation waves. The excitation waves are physically mapped onto the hardware actuation waves. We analyse different types of excitation and manipulation patterns and physical implementations of the actuating surface.

13.1 Introduction

Traditionally industrial manipulation tasks are performed in an autonomous manner with the use of large, usually 6 Degrees of Freedom manipulators, that can move objects lighter than the manipulating robot itself. Such manipulators are either single units or small groups of units. In the groups of units collaborative tasks are achieved by planning the spatial trajectories of individual units in advance, often before start of the manipulation. Although the vast majority of manipulation tasks are still completed by these robots a new type of manipulation emerged: micro-scale manipulation and assembly. This new type of manipulation require high precision, optimum force application, non-prehensile handling and concurrent manipulation of several objects during the work cycle. Classic manipulation approaches struggle to satisfy these requirements. We therefore explore a specialised massively parallel hardware operating like a smart manipulating surface [13]. A manipulating surface is

I. Georgilas (✉) · A. Adamatzky · C. Melhuish
University of the West of England, Coldharbour Lane, Bristol, UK
e-mail: ioannis.georgilas@uwe.ac.uk

C. Melhuish
Bristol Robotics Laboratory, Coldharbour Lane, Bristol, UK
e-mail: Chris.Melhuish@brl.ac.uk

A. Adamatzky
e-mail: andrew.adamatzky@uwe.ac.uk

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4_13

an array of simple actuators, each of them has a small power output, that collectively transport, orient and position objects whose masses and sizes are higher compared to the power output generated by a single actuator. Each individual actuator is relatively inexpensive, and a modular structure of the parallel manipulator would allow for mass production and scalability.

One of the key technologies utilised to control the motion of the objects for these systems are arrays of air jets [11, 14, 22, 24]. Other approaches include mechanical wheel based arrangements [23, 26], sound based solutions [31] and electromechanical actuators to excite membranes [12, 27]. Most of these manipulation methods are using the underling physical phenomena in an intelligent way to control the manipulation, i.e. flow interactions [16], and anisotropic friction control [30]. Various control systems have been proposed mainly on the concept of closed-loop control, with feedback provided either from a camera [15] or other form of light information, i.e. photodiodes [11].

The main issue in all these control methods is a scalability of the task. Controlling the vast number of actuators is computationally plausible as shown by simulation examples [17] but still expensive and usually comes at the cost of precision [8]. The manipulation task becomes even more complicated when multiple objects must be processed simultaneously and the controller must synchronise the spatio-temporal trajectories of the objects.

A strong alternative to address the problem of scalability of control is the use of lattice automata as the underlying controllers of the manipulation array. Specifically each individual actuator's state is controlled by the state of the automaton. The intelligence and control is achieved by the emergence of order in the lattice. Furthermore the system scales uniformly since the computation cost of the state-machine for each actuator remains the same irrespective of the number of actuators. The use of lattice automata as robot controllers is not new and have been utilised to control path planning for mobile robot platforms [4, 5, 29]. The proposed here use of the lattice automata differentiates from previous implementations in the effort to enhance and bring forward the synergies necessary to complete the required manipulation task.

We discuss firstly what characteristics of lattice automata are ideal for this type of control, and what 'modes' of operations optimally can achieve this, and by which set of state rules. Also some details on the importance of the hardware and how it improves the synergy will be given. We evaluate the behaviour and operation of the lattice performance in a series of simulation and real-world experiments. The qualitative and quantitative performance of the experimental prototype of cellular-automaton actuator array is analysed.

13.2 Cellular Automata Controller

Actuators arranged in a two-dimensional array is one of most optimal ways to achieve modularity, controllability and fault tolerance. If we want an actuator array to be autonomous, i.e. not controlled by a host computer, we should allow each elementary

actuating unit to have some computing power, in principle some local sensing, and be able to interact with its immediate actuating neighbours. By using local interaction the actuators can establish a coordinated action and manipulate objects, which are substantially larger in size than a single actuator, leading to epiphenomenal 'task' behaviour. Cellular automata would be ideal controller for the arrays of actuators. This is because a cellular automaton is an array of finite state machines, or cells, which update their states in discrete time depending on states of their immediate neighbours. Thus we can assign a unique cell of a cellular automaton to an elementary unit of an actuating array. The topology is preserved. All cells of a cellular automaton update their states by the same rule, all units of the actuator array act by the same mechanics.

A requirement for focused force application suggests us that most convenient manipulation rules should be based on propagating patterns, either omni-directional waves or travelling localizations [3]. Mobile self-localizations can be described as waves or wave-fragments (gliders, wavelets), the manipulation abilities of which have been previously demonstrated in [21]. The use of wave-fragments and their synchronisation signals have been analysed in [18], where the concept of metachronal waves [7] was applied in lattice automata-controlled hardware.

The lattice automaton we are proposing to utilise is the 2^+ medium [1, 6], a 3-state (excited (+), refractory (−), resting(·)) cellular automaton with well defined mobile self-localizations. The cell-state transition rule of the 2^+ medium is given in Eq. 13.1: a resting cell excites if it has exactly two excited neighbours; an excited cells becomes refractory; a refractory cell returns to a resting state.

$$x^{t+1} = \begin{cases} +, & x^t = \cdot \text{ and } \sum_{y \in u(x)} \chi(y, +) = 2 \\ -, & x^t = + \\ \cdot, & \text{otherwise} \end{cases} \tag{13.1}$$

where $\chi(y, +) = 1$ if $y = +$ and 0, otherwise.

In the 2^+ medium both omnidirectional waves and wave-fragments can be created using simple initial conditions. Both types of waves can travel in the cardinal directions, which is an extra benefit for manipulation tasks, since it allows for clear directional vectors for the manipulated objects. The basic manipulation element is the wave-fragment, or glider, consisting of an excited head (two excited cells) and a refractory tale (two refractory cells), Fig. 13.1 The ternary nature of the automaton is convenient for the hardware interface. Each state of the automaton can be directly mapped onto a state of actuator motor: off, clockwise spin, and counter-clockwise spin.

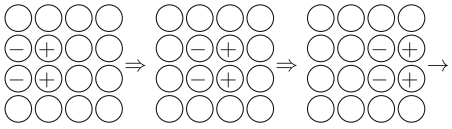


Fig. 13.1 Typical travelling localisation in 2^+-medium [2]. The localisation propagates eastward. The excitation wave-front is followed by refractory tail. Excited sites are shown by '+', refractory by '−'

Table 13.1 Combination scenarios of physical contacts between actuator surfaces and manipulated object

Cilia in...	Scenario 1	Scenario 2	Scenario 3	Scenario 4[a]
Object	−	+	−	+
Surface	−	−	+	+

Presence of cilia on object or surface is indicated by '+', absence by '−'
[a]Scenario 4 has not been investigated

13.3 Hardware Layer Role

The hardware architecture of a cellular automaton controller is proposed in [20]. The architecture draws its inspiration from the ciliary motion of *Paramecium caudatum*. Cilia are also used to exploit anisotropic friction [25, 32] and thus to generate force fields to move the object. Although some research teams have introduced this idea of programable force-fields with anisotropic friction using vibratory motion [28, 30], our method differentiates on the mechanism of generating the friction, hence the force. We propose the coordinated vibratory motion of cilia structures controlled by lattice automata.

To fully evaluate the applicability of the proposed method different hardware architectures are investigated. The differentiation factor of these architectures is the location of the cilia-like structures. Table 13.1 gives us the potential combinations.

Scenario 4 in Table 13.1 is being presented only for completeness. Although it is possible to create a system with cilia in both the object and the manipulation surface, it will be a system of complex physical interactions between the cilia on actuators' surfaces and the cilia on the manipulated object.

13.4 Experimental and Simulation Results

Each of the scenarios presented in Table 13.1, except Scenario 4, was tested in hardware or computer simulation to evaluate the performance of the lattice automata in control. In hardware tests we used the prototype described in [19, 20]. The prototype is an array 8×8 oscillating motors, covered with a silicon membrane. The array is 110×110 mm in size. Rotation direction of each motor is controlled independently on other motors. We used overhead video camera to measure motion of the manipulated objects.

Computer simulation of the actuator array was made using MATLAB and APRON—(A)rray (P)rocessing envi(RON)ment [9], a real-time simulation platform for working with and debugging two-dimensional arrays of data and rapidly prototyping array based algorithms. For some of the simulation experiments, where the focus was on the physics-based analysis of manipulation, an environment for physics simulation was utilised [10].

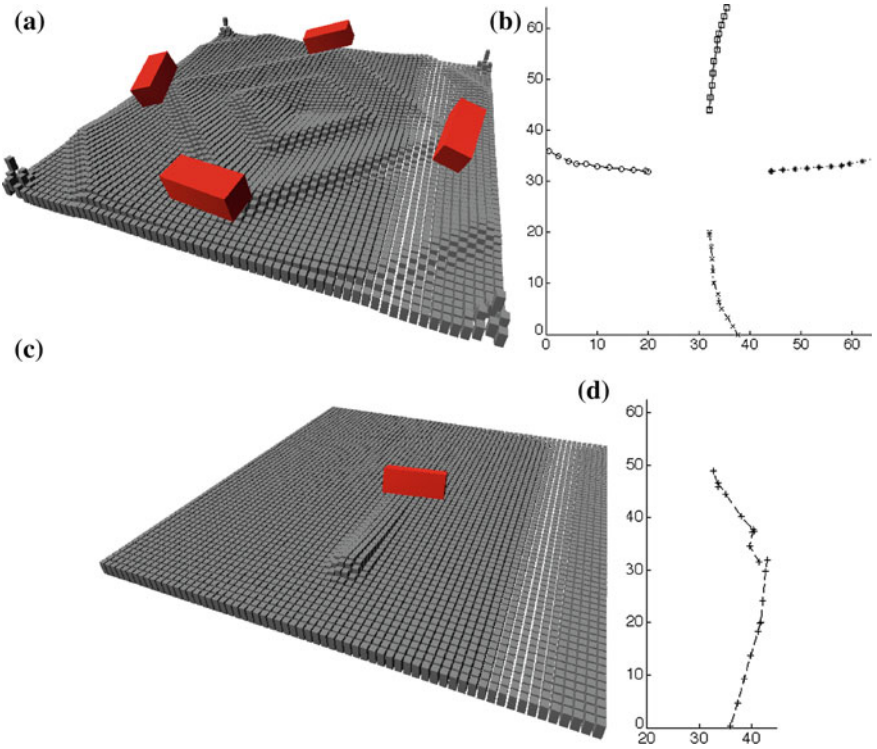


Fig. 13.2 Manipulation with waves. Simulation frames of manipulating objects with omni-directional wave (**a**) and travelling localisation (**c**). Trajectories of objects translated by these waves are shown in (**b**) and (**d**). x and y-axis in simulator distance units. Simulation is implemented in APRON [9]

13.4.1 Scenario 1: No Cilia

In the no-cilia scenario two sets of simulation experiments were carried out and the behaviour of omni-directional and wave fragments was investigated. In the initial experiment, firstly four objects were displaced using omni-directional waves, Fig. 13.2a, and secondly a single object using linearly propagating wavelets, Fig. 13.2c. In the latter experiment only one object was used given the narrow focus of the glider. The trajectories of the objects were recorded. They are shown in Fig. 13.2b, d.

In the laboratory experiments with hardware prototype the object shown in Fig. 13.3a was manipulated by the actuator array. The object is a 40×20 mm hollow, plastic, rectangular box. We did not execute laboratory experiments with omni-directional waves because in such type of manipulation objects move along coupled trajectories which is not suitable for a precise manipulation; also, the manipulated objects 'hopped' above the surface, see more details in Sect. 13.5. In the laboratory

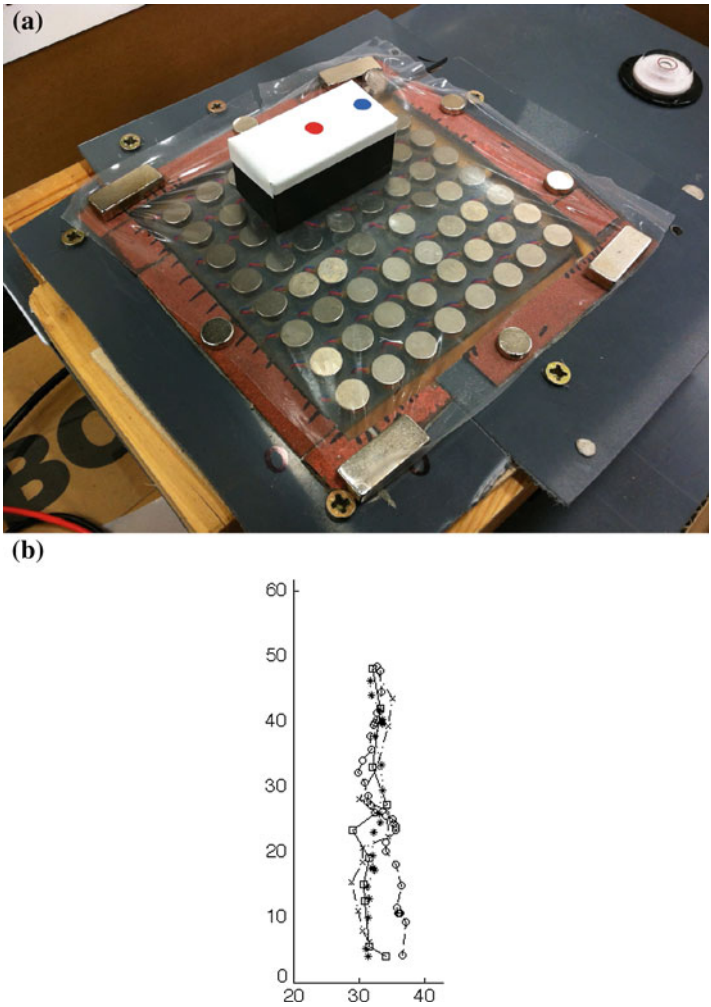


Fig. 13.3 Neither actuator array nor the manipulated object are equipped with cilia. **a** Photograph of a hardware prototype of 8×8 actuator array with the manipulated object on *top*. **b** Four experimental trajectories; x and y axises adjusted to simulator distance units

experiments on manipulating with travelling localisations four different trajectories were recorded, see Fig. 13.3b. Although each trajectory is slightly different due to mechanical variations of the prototype, the overall pattern is as predicted in the simulation experiment.

As can be observed for the omni-directional waves the 'broadcast' mode is typical: waves traveling all across the lattice render the trajectories of the objects coupled. Moreover, as seen from the screen shot in Fig. 13.2a, the delivered energy causes vertical displacement of the objects, a 'hopping' effect. On the other hand, the wavelet

manipulation operates as expected, the object moves in a linear fashion. The slight parabolic motion can be attributed to the specific dynamics of the simulation engine.

13.4.2 Scenario 2: Object with Cilia/Surface Without Cilia

In the second scenario an object with cilia was manipulated by the actuator array without cilia. We selected a toothbrush as a manipulated object due to its intriguing geometry, see Fig. 13.4a, b and the availability of a ciliated object of standardised construction. The toothbrush was 35 × 12 mm in size. A photo of the experimental prototype is shown in Fig. 13.4c. The cilia, or bristles, of the toothbrush create an anisotropic friction that allows the surface to manipulate the object.

Two sets of experiments have been conducted. The first set dealt with linear motion of the toothbrush, the target is in the same line as the toothbrush's initial position. In the second set of experiments, we introduced a change of the toothbrush's direction path is necessary: the target was positioned at a 90° angle from the line of direction of the toothbrush. This cornering action was achieved by a combination of glider motions and turning patterns. The turning patterns were represented in cellular automaton controller as two excited + and two refractory − cells in a crossed configuration.

Two trajectories were recorded for each set. In Figs. 13.5 and 13.6 the overlap of the trajectories, with rotation tracked by the directional vectors, can be seen along with the start position of the toothbrush and the target. In Figs. 13.5 and 13.6 the trajectory and rotation of toothbrush body is separately plotted for ease of analysis.

By analysing the data of the trajectories we can identify some interesting patterns. Although there are variations, both sets of trajectories are similar with most of the same characteristic motifs. These variations can be attributed to specific hardware and software aspects. Three key aspects are identified:

- **Drifting movement of the toothbrush.** The toothbrush head, because of its special geometry, tends to move in a drifting fashion, oscillating from side to side.
- **Motor array variations.** Although the highest specifications where used to design and construct the prototype some variations still exist. Differences in geometry of the modules and inevitable differences in motor operation (manufacturer's specifications) result in small variations of performance from cell to cell.
- **Algorithm and CA propagation.** The implementation of the lattice automaton algorithm affects operations in two distinctive ways. First, the pattern used to turn the toothbrush might affect its trajectory in an unwanted manner. Secondly, the lattice refresh interval (200 m in the experiments) affects movement patterns, indicating that there are other spatio-temporal dynamic phenomena at play impacting the operation of the system.

The different reasons for the patterns recognised in the experiments, allow to draw some interesting experience regarding the system both in hardware and software terms. The first two, drifting movement and motor variations, are related to the

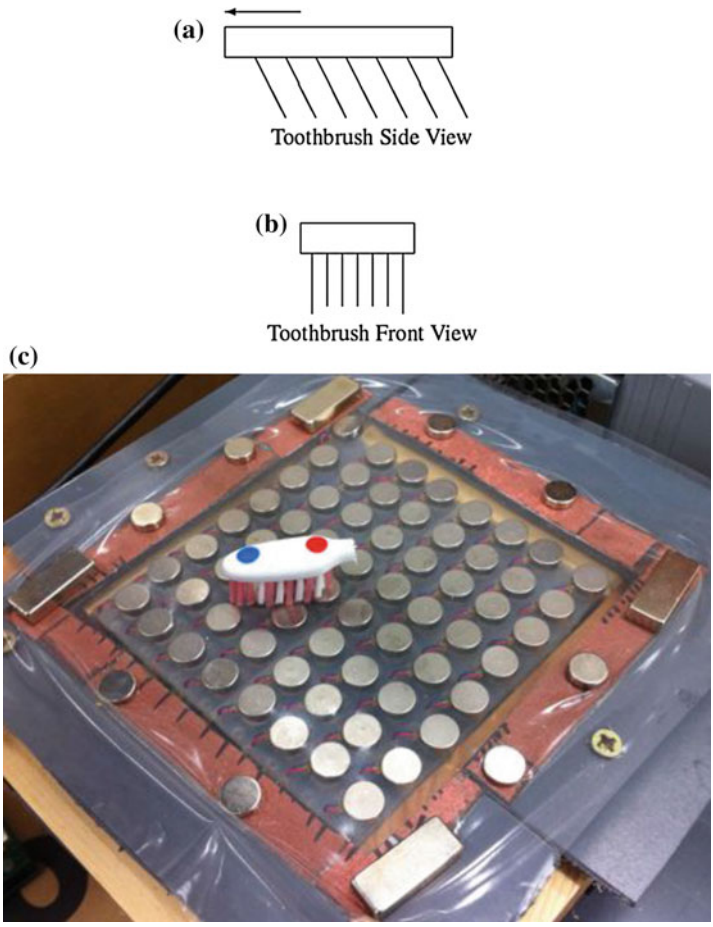


Fig. 13.4 Manipulated object has cilia but actuator array has none. **a** Side and **b** front view of object's cilia geometry responsible for the anisotropic friction, **c** Photograph of the parallel manipulator with an ciliated object on *top*

hardware being used, toothbrush head geometry and prototype variant. In order to emphasise the robustness of the proposed conveyor system the decision was taken to accept mechanical variations within specific tolerances. This way the system proves that it can compensate for those variations using the intelligent underlying control algorithm. Due to the systemic occurrences of both phenomena, it is possible to map those mechanical imperfections and incorporate them in the lattice automata controller making the system more robust.

The third reason is related to the implementation of the automaton controller. Simple linear gliders in 2^+ medium seem to provide a good linear object propagation while the proposed turning patterns address to a certain degree the 90° turns required for the selected trajectories. Both the gliding and the turning patterns perform

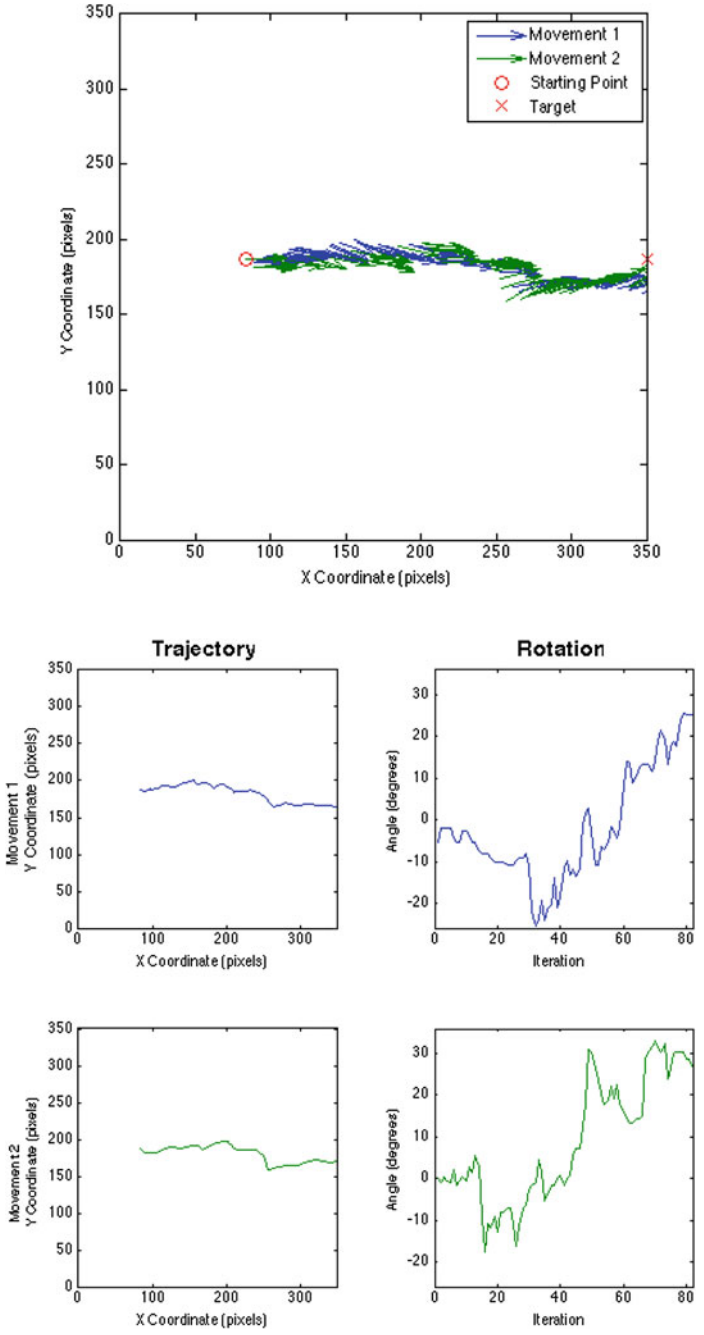


Fig. 13.5 Set of Linear Movements

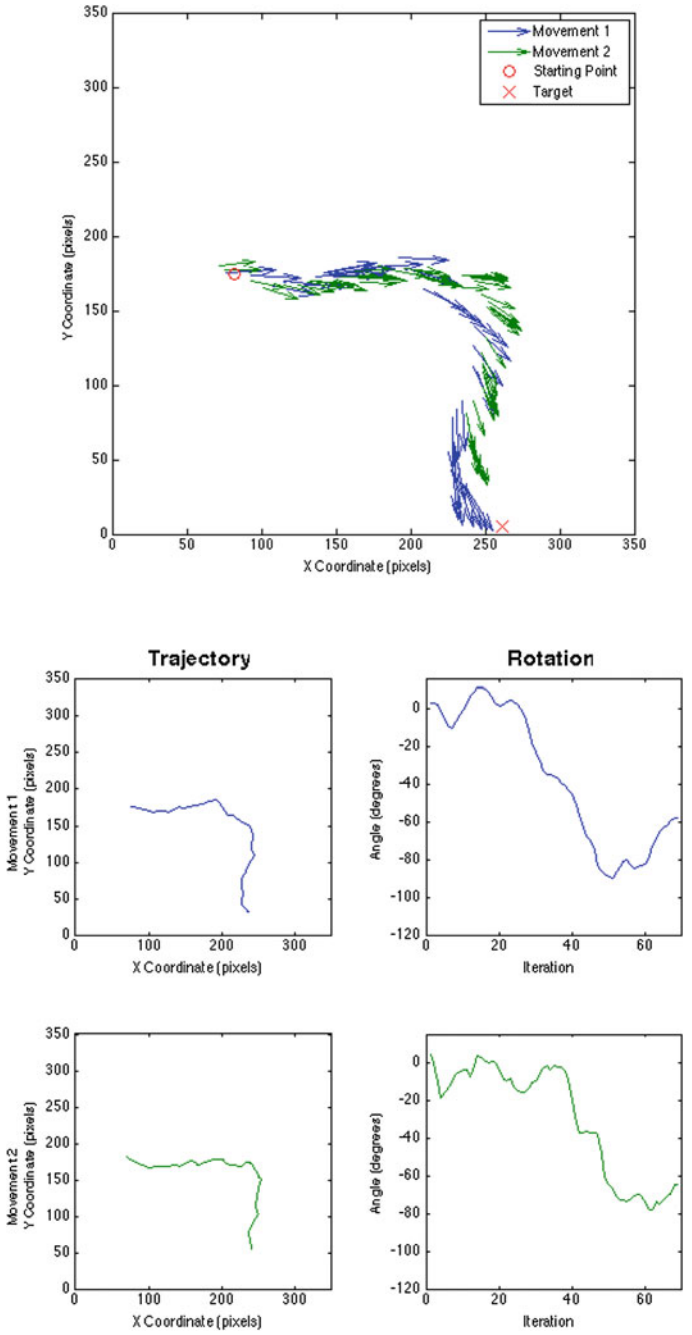


Fig. 13.6 Set of Corner (90°) Movements

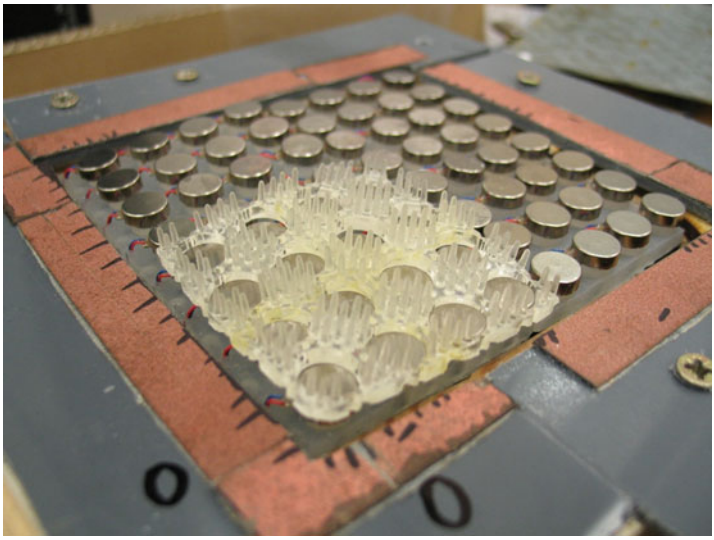


Fig. 13.7 Photograph of the bristles fabricated to emulate natural cilia. This design is used as an alternative to the membrane design described in [20]

satisfactorily in compensating for the hardware's weaknesses. Undoubtedly, determining the iteration (generation) step is a crucial parameter affected by the dynamics of the system.

13.4.3 Scenario 3: Object Without Cilia/Surface with Cilia

The third scenario is tested using simulation experiments based on the modelled behaviour of the ciliated surface. A small hardware version of the proposed system has been fabricated for verification, see Fig. 13.7. The model is based in the vortex-like behaviour of a single motor and the resulting force field as exerted by the cilia structures is shown in Fig. 13.8a.

In this scenario, the task was to move a square object from the lower right corner of the lattice to the centre of it. This corresponds to the natural task encountered in microorganisms for the transportation of food to their 'mouth' pore [18]. The importance of synchronisation signals was investigated with the application of three different signals. Specifically the signals tested are (a) Single motor actuation, Fig. 13.8b; (b) Random motor actuation, Fig. 13.8c; and (c) Metachronal wave motor actuation, created by linearly traveling localisation, Fig. 13.8d, f. This selection of signal was made to demonstrate the necessity of proper synchronisation and co-action of actuators in the manipulation tasks.

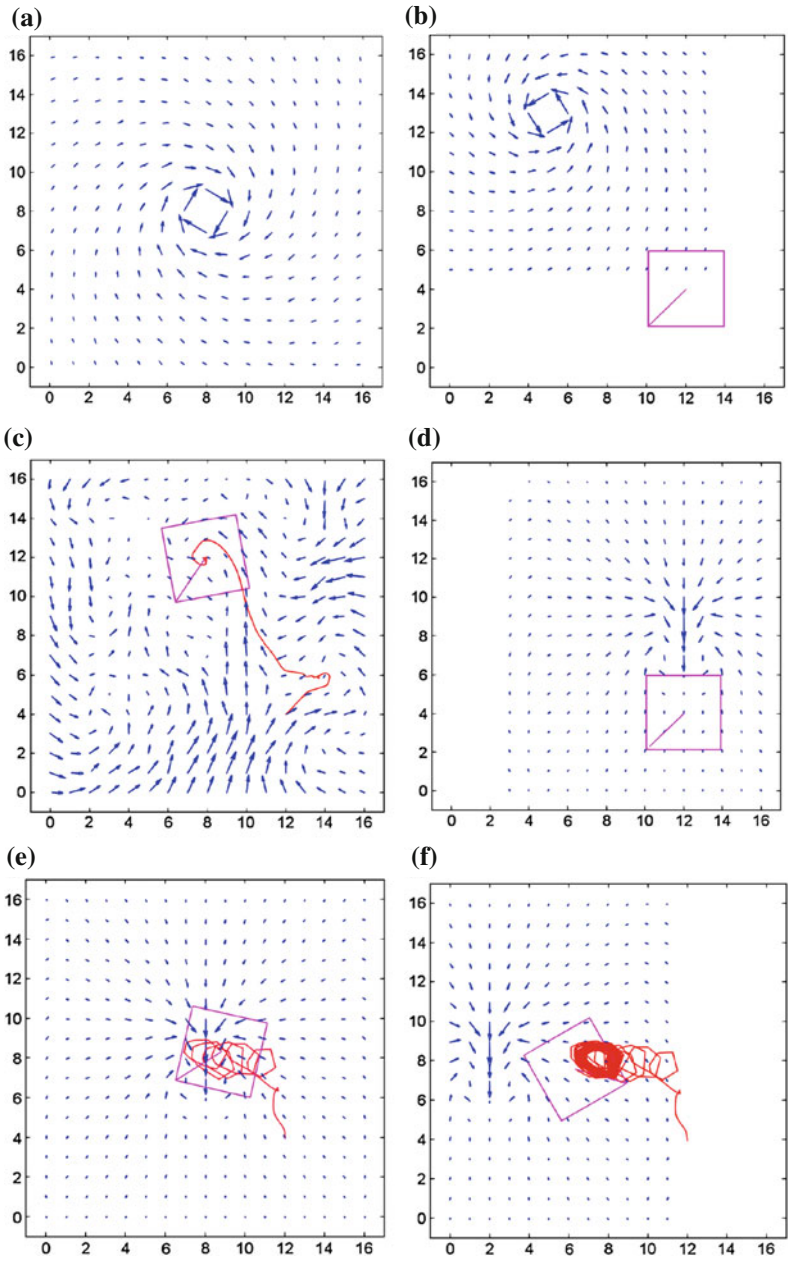


Fig. 13.8 Simulation frames from MATLAB with APRON generated control signals. The vectors of the force field, the rectangular object with a rotation indication line and the trajectory are depicted. **a** The vortex force-field created by a single motor. **b** The single motor force field 'pulling' the object towards the centre of the lattice (frame 60). **c** The trajectory of the object under a randomly generated force field (frame 400). **d**–**f** Frames 0, 60 and 400 of the trajectory under the metachronal wave signal. Object is placed at coordinates (12,4). Axis are simulation based units

Analysing the trajectories from the different control signals we can deduce some interesting results regarding the cooperative attributes of the various control signalling (Fig. 13.8). In the single motor approach, the motor trying to 'pull' the object towards the centre fails even to start moving the object (Fig. 13.8a, b). Investigating the exhibited forces we find that the single motor is not able to exert the friction between the object and the surface. This is a demonstration that a single 'cilium' fails to achieve the task and some form of cooperative action needs to take place. The random motor approach overcomes the lack of power, since it does move the object from its initial position. Nonetheless, the object is moving along a random path (Fig. 13.8c). Furthermore, it might be locked in local attractors that will not coincide with the intended target. Hence, the use of multiple 'cilia' is necessary to produce manipulation, but random 'beating', i.e. signalling, does not create controllable behaviour. Finally, the 2^+-medium glider, seen as metachronal wave, moves the object towards the target (Fig. 13.8d, f).

13.5 Conclusions

In this chapter we demonstrate the use of lattice automata as distributed control systems for manipulation applications. We have shown that controllable manipulation can be reached via interaction between cells/actuators and also between physical surfaces of the actuator units and the manipulated object. In all three scenarios, as shown in computer modelling and laboratory experiments, travelling localisations (gliders, solitons, wave-fragments) are proved to be ideal manipulating patterns. Further studies are required to select the best morphology of the control patterns in cellular automata, which can lead to robust and precise manipulation. Actuator arrays discussed in the chapter are open-loop manipulators: actuating units do not sense the manipulated object. In our future research we aim to develop closed-loop manipulators, where actuators are equipped with sensors allowing them to feel the manipulated object.

References

1. Adamatzky, A.: Dynamical universal computation in excitable lattices. In: Margenstern, M. (ed.) MCU, vol. 2, pp. 194–213. IUT, Metz (1998)
2. Adamatzky, A.: Computing in Non-linear Media and Automata Collectives. Institute of Physics Publishing, Bristol (2001)
3. Adamatzky, A., Costello, B.D.L., Asai, T.: Reaction-Diffusion Computers. Elsevier, New York (2005)
4. Adamatzky, A., De Lacy Costello, B., Melhuish, C., Ratcliffe, N.: Experimental reaction-diffusion chemical processors for robot path planning. J. Intel. Robot Syst. **37**(3), 233–249 (2003). doi:10.1023/A:1025414424756
5. Adamatzky, A., Melhuish, C.: Phototaxis of mobile excitable lattices. Chaos. Soliton. Fract. **13**(1), 171–184 (2002). doi:10.1016/S0960-0779(00)00233-2

6. Adamatzky, A.I.: Controllable transmission of information in excitable media: The 2+ medium. Adv. Mater. Opt. Electron. **5**(3), 145–155 (1995). doi:10.1002/amo.860050303, http://dx.doi.org/10.1002/amo.860050303
7. Aiello, E., Sleigh, M.A.: The metachronal wave of lateral cilia of mytilus edulis. J. Cell Biol. **54**(3), 493–506 (1972)
8. Ashley-Rollman, M., Pillai, P., Goodstein, M.: Simulating multi-million-robot ensembles. In: Proceedings—IEEE International Conference on Robotics and Automation, pp. 1006–1013 (2011)
9. Barr, D.R., Dudek, P.: Apron: a cellular processor array simulation and hardware design tool. EURASIP J. Adv. Signal Process. **2009**, 9 (2009)
10. Barr, D.R., Walsh, D., Dudek, P.: A smart surface simulation environment. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013, pp. 4456–4461. IEEE (2013)
11. Berlin, A., Biegelsen, D., Cheung, P., Fromherz, M., Goldberg, D., Jackson, W., Preas, B., Reich, J., Swartz, L.E.: Motion control of planar objects using large-area arrays of mems-like distributed manipulators. Xerox Palo Alto Research Center CA/USA, Presented at Micromechatronics (2000)
12. Bohringer, K.F., Bhatt, V., Donald, B., Goldberg, K.: Algorithms for sensorless manipulation using a vibrating surface. Algorithmica **26**(3–4), 389–429 (2000). (New York)
13. Bohringer, K.F., Donald, B.R., Mihailovich, R., MacDonald, N.C.: Sensorless manipulation using massively parallel microfabricated actuator arrays. In: Proceedings on IEEE International Conference on Robotics and Automation, 1994, pp. 826–833. IEEE (1994)
14. Boutoustous, K., Laurent, G., Dedu, E., Matignon, L., Bourgeois, J., Le Fort-Piat, N.: Distributed control architecture for smart surfaces. In: 2010 IEEE/RSJ International Conference Intelligent Robots and Systems, IROS'10, pp. 2018–2024, Taipei (2010)
15. Delettre, A., Laurent, G., Le Fort-Piat, N.: 2-dof contactless distributed manipulation using superposition of induced air flows. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'11, pp. 5121–5126. San Francisco, CA (2011). doi:10.1109/IROS.2011.6048251
16. Delettre, A., Laurent, G.J., Fort-Piat, L., Varnier, C., et al.: 3-dof potential air flow manipulation by inverse modeling control. In: IEEE International Conference on Automation Science and Engineering (CASE), 2012, pp. 930–935. IEEE (2012)
17. Dhoutaut, D., Piranda, B., Bourgeois, J.: Efficient simulation of distributed sensing and control environments. In: Proceedings—2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013, pp. 452–459 (2013)
18. Georgilas, I., Adamatzky, A., Barr, D., Dudek, P., Melhuish, C.: Metachronal waves in cellular automata: Cilia-like manipulation in actuator arrays. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), pp. 261–271. Springer International Publishing (2014)
19. Georgilas, I., Adamatzky, A., Melhuish, C.: Manipulating objects with gliders in cellular automata. In: 2012 IEEE International Conference on Automation Science and Engineering (CASE), pp. 936–941. IEEE (2012)
20. Georgilas, I., Adamatzky, A., Melhuish, C.: Towards an intelligent distributed conveyor. In: Advances in Autonomous Robotics pp. 457–458 (2012)
21. Georgilas, I., Adamatzky, A., Melhuish, C.: Manipulating with excitations: Waves or gliders? In: Workshop notes of the ICRA Workshop in Unconventional Approaches to Robotics, Automation and Control Inspired by Nature, vol. International Conference in Robotics and Automation (ICRA). Karlruhe (2013)
22. Laurent, G.J., Delettre, A., Zeggari, R., Yahiaoui, R., Manceau, J.F., Fort-Piat, N.L.: Micropositioning and fast transport using a contactless micro-conveyor. Micromachines **5**(1), 66–80 (2014). doi:10.3390/mi5010066, http://www.mdpi.com/2072-666X/5/1/66
23. Luntz, J., Messner, W., Choset, H.: Distributed manipulation using discrete actuator arrays. Int. J. Rob. Res. **20**(7), 553–583 (2001)

24. Moon, H., Luntz, J.: Distributed manipulation of flat objects with two airflow sinks. IEEE Trans. Robot. **22**(6), 1189–1201 (2006). doi:10.1109/TRO.2006.882921
25. Mróz, Z., Stupkiewicz, S.: An anisotropic friction and wear model. Int. J. Solids Struct. **31**(8), 1113–1131 (1994)
26. Murphey, T., Burdick, J.: Feedback control methods for distributed manipulation systems that involve mechanical contacts. Int. J. Robot Res. **23**(7–8), 763–781 (2004)
27. Setter, E., Bucher, I.: Flexural vibration patterning using an array of actuators. J. Sound. Vib. **330**(6), 1121–1140 (2011)
28. Umbanhowar, P., Vose, T., Mitani, A., Hirai, S., Lynch, K.: The effect of anisotropic friction on vibratory velocity fields. In: Proccedings of 2012 IEEE International Conference on Robotics and Automation, pp. 2584–2591 (2012)
29. Vazquez-Otero, A., Faigl, J., Munuzuri, A.: Path planning based on reaction-diffusion process. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'12, pp. 896–901 (2012)
30. Vose, T., Turpin, M., Dames, P., Umbanhowar, P., Lynch, K.: Modeling, design, and control of 6-dof flexure-based parallel mechanisms for vibratory manipulation. Mech. Mach. Theory **64**, 111–130 (2013)
31. Vose, T., Umbanhowar, P., Lynch, K.: Vibration-induced frictional force fields on a rigid plate. In: 2007 IEEE International Conference on Robotics, ICRA'07, pp. 660–667. Rome (2007)
32. Zmitrowicz, A.: Mathematical descriptions of anisotropic friction. Int. J. Solids Struct. **25**(8), 837–862 (1989)

Index

G.Ch. Sirakoulis and A. Adamatzky (eds.), *Robots and Lattice Automata*,
Emergence, Complexity and Computation 13, DOI 10.1007/978-3-319-10924-4