Springer Tracts in Advanced Robotics 112

Nak-Young Chong
Young-Jo Cho *Editors*

Distributed Autonomous Robotic Systems

The 12th International Symposium

Springer Tracts in Advanced Robotics 112

Editors

Prof. Bruno Siciliano
Dipartimento di Ingegneria Elettrica
e Tecnologie dell'Informazione
Università degli Studi di Napoli
Federico II
Via Claudio 21, 80125 Napoli
Italy
E-mail: siciliano@unina.it

Prof. Oussama Khatib
Artificial Intelligence Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305-9010
USA
E-mail: khatib@cs.stanford.edu

Nak-Young Chong · Young-Jo Cho
Editors

Distributed Autonomous Robotic Systems

The 12th International Symposium

Springer

Editors
Nak-Young Chong
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa
Japan

Young-Jo Cho
Electronics and Telecommunications Research Institute
Daejeon
Korea, Republic of (South Korea)

ISSN 1610-7438 ISSN 1610-742X (electronic)
Springer Tracts in Advanced Robotics
ISBN 978-4-431-55877-4 ISBN 978-4-431-55879-8 (eBook)
DOI 10.1007/978-4-431-55879-8

Library of Congress Control Number: 2015957775

Printed on acid-free paper

This Springer imprint is published by SpringerNature
The registered company is Springer Japan KK

Series Foreword

Robotics is undergoing a major transformation in scope and dimension. From a largely dominant industrial focus, robotics is rapidly expanding into human environments and vigorously engaged in its new challenges. Interacting with, assisting, serving, and exploring with humans, the emerging robots will increasingly touch people and their lives.

Beyond its impact on physical robots, the body of knowledge robotics has produced is revealing a much wider range of applications reaching across diverse research areas and scientific disciplines: biomechanics, haptics, neurosciences, virtual simulation, animation, surgery, and sensor networks among others. In return, the challenges of the new emerging areas are proving an abundant source of stimulation and insights for the field of robotics. It is indeed at the intersection of disciplines that the most striking advances happen.

The *Springer Tracts in Advanced Robotics* (STAR) is devoted to bringing to the research community the latest advances in the robotics field on the basis of their significance and quality. Through a wide and timely dissemination of critical research developments in robotics, our objective with this series is to promote more exchanges and collaborations among the researchers in the community and contribute to further advancements in this rapidly growing field.

DARS is a well-established single-track conference that gathers every 2 years the main researchers in *Distributed Autonomous Robotic Systems*. Since the 10th edition in 2010, STAR has welcomed DARS among the volumes resulting from thematic symposia devoted to excellence in robotics research.

The volume edited by Nak Young Chong and Young-Jo Cho offers in its 32 chapters an interdisciplinary collection of technologies, algorithms, system architectures, and applications of advanced distributed robotic systems. The contents are effectively grouped into four thematic sections: collaborative exploration, localization, and mapping; cooperative manipulation and task allocation; formation control and path planning; multi-robot communication and control architecture.

Rich by topics and authoritative contributors, the 12th edition of DARS in 2014 culminates with this unique reference on the current developments and new directions in the field of distributed autonomous robotic systems. A very fine addition to STAR!

Naples, Italy
August 2015

Bruno Siciliano
STAR Editor

Preface

The latest volume in the Distributed Autonomous Robotic Systems series constitutes the thoroughly reviewed post-conference proceedings of the 12th International Symposium on Distributed Autonomous Robotic Systems (DARS 2014), which was held at the Daejeon Convention Center, Daejeon, Korea, November 2–5, 2014. Following the tradition established by the previous symposiums since 1992, the goal of DARS 2014 has been to exchange and stimulate research ideas to realize advanced distributed robotic systems. Distributed robotics is a rapidly growing, interdisciplinary research area lying at the intersection of computer science, communication and control systems, and electrical and mechanical engineering. Stunning examples of cutting-edge technologies, algorithms, system architectures, and applications were presented and discussed during a single-track, three-day symposium. Building on the momentum of successful previous symposiums, DARS 2014 also provided a supportive and exciting environment for academics and practitioners to present and discuss their novel theoretical results, implementations, and applications in distributed autonomous robotic systems.

DARS 2014 received a total of 81 papers from 11 countries. Sixty papers were submitted to the regular paper track and 21 papers were submitted to the work-in-progress paper track. For the first time in the history of DARS, the work-in-progress poster session was designed to allow authors to present new challenges and directions, and early and emerging results from both academia and industry, providing a forum for the discussion of timely topics and promising yet still-undeveloped ideas. This effort will further push the boundaries of our scientific and technical limits and expand the horizons of DARS beyond academia. The final technical program consists of 29 papers in a total of nine oral sessions and 25 papers in one poster session. The oral session provides a platform for authors to present and discuss their new findings and controversies in a formal way within a 20-minute time slot. The 80-minute poster session allows authors to facilitate more personal interactions with more targeted and interested audiences, and affords more time to present their work in depth.

Finally, 32 papers of the highest quality, carefully selected and revised after the symposium, are included in this volume. These papers will give a broad, yet focused perspective categorized into the following four areas: (1) collaborative exploration, localization, and mapping, (2) cooperative manipulation and task allocation, (3) formation control and path planning, and (4) multi-robot communication and control architecture. Well-defined specific research problems in the respective topic areas are investigated and analyzed on theoretical grounds and experimental confirmation under real-world conditions. Specifically, this volume elaborates on "distributed autonomy" that is efficient and scalable compared to the best-known centralized algorithms in the literature, and envisions ways it can evolve to be more sustainable. The latest findings and implications learned from all of the above-mentioned areas will help readers understand how and why various forms of cooperative interactions emerge and flourish in distributed autonomous robotic systems, and push them into today's demanding applications and large-scale distributed systems. This volume will be of great use to postgraduate students, researchers, and practitioners wishing to study a range of current and emerging issues and specific topics in distributed autonomous robotic systems.

In addition to ten regular sessions centered on distributed autonomy themes, DARS 2014 was honored to have four distinguished plenary session speakers. The titles and abstracts of the plenary lectures are given below:

Distributed Systems for Urban Mobility

Professor Emilio Frazzoli, Massachusetts Institute of Technology, USA

The first part of this talk will concentrate on self-driving cars, and their impact on personal mobility in urban settings. Research and development on self-driving cars is currently very active, and cars able to drive safely and reliably without need for human supervision are no longer science fiction. Indeed, several companies and universities have demonstrated vehicles able to drive autonomously in traffic, in the process building social awareness and pushing the boundaries of current regulations and risk management practices. At this point, a natural question to ask is: what is the point of autonomous cars? Is autonomy indeed a transformative technology, with a potential to drastically redefine mobility? If so, in what ways, and when?

I will argue that the "killer app" for self-driving cars is car sharing, and will provide analytical guidelines and financial justification for the design of shared-vehicle mobility-on-demand systems. As a case study, we consider replacing all modes of personal transportation in a city such as Singapore with a fleet of shared automated vehicles, able to drive themselves, e.g., to move to a customer's location. Using actual transportation data, our analysis suggests a shared-vehicle mobility solution can meet the personal mobility needs of the entire population with a fleet whose size is approximately one-third of the total number of passenger vehicles currently in operation.

The second part of the talk will concentrate on distributed algorithms for traffic signal control. The proposed algorithms are adapted from backpressure routing, which has been mainly applied to communication and power networks. Our algorithm ensures global optimality as it leads to maximum network throughput even though the controller is constructed and implemented in a completely distributed manner. Simulation results show that our algorithm significantly outperforms state-of-the-art algorithms.

Multi-robot Collision Avoidance and Applications

Professor Beom Hee Lee, Seoul National University, Korea

Nowadays, multi-robot operations are acknowledged as a common practice in industry for various tasks. The state of the art of multi-robot systems is described in the first statement. Multi-robot research issues are then discussed in terms of the operational strategies: centralized, distributed, and mixed operational schemes. Next, we show that one of the main issues in multi-robot operation is the problem of collision avoidance. We also show the importance of the collision avoidance problem in multi-robot operations. For multi-robot collision avoidance, a special tool, called the collision map, is introduced and applied to this problem. More deep analysis and investigation are presented for an application of the collision map. Various types of collision maps are then introduced with several possible applications. Also, robot path modification is viewed in terms of collision avoidance using the concept of the collision map. Various applications using the collision map are presented for a problem of 100 multi-robot operations, a stealth intruder intercept scheme, and efficient multiple cleaning robots operation. Especially, the load balancing in multiple cleaning robots are realized using the collision map. Finally, future applications using multi-robot systems are briefly discussed.

Design and Navigation of Robots that Roll, Run, and Fly

Professor Roland Siegwart, ETH Zurich, Switzerland

Robots are rapidly evolving from factory workhorses, which are physically bound to their work-cells, to increasingly complex machines capable of performing challenging tasks as search and rescuing, surveillance and inspections, planetary exploration or autonomous transportation of goods. This requires robots to operate in unstructured and unpredictable environments and various terrains. This talk will focus on design and navigation aspects of wheeled, legged, swimming and aerial robots operating in complex environments. Our wheeled robots are designed to move on complex grounds or to autonomously drive in parking lots. For our quadruped walker we are researching optimal ways to exploit the natural dynamics and serial elastic actuation. Our swimming robots take inspiration from natural

counterparts for optimal propulsion, and with our micro-helicopter projects we approach autonomous flights and inspections in cluttered and very narrow indoor environments as well as GPS-denied visual navigation in cities. And our small solar airplanes are capable of staying in the air indefinitely and flying close to the ground thanks to onboard vision.

A Synchronization Control Approach to Networked Robotic Systems

Professor Dong Sun, City University of Hong Kong, Hong Kong, China

Nowadays cooperative controls of networked robotic systems have become a hot research area with dramatically increased popularity. Synchronization is a common timekeeping methodology which requires the coordination of events to operate a system in unison. This talk will introduce our researches of using synchronization control approach to motion coordination of networked robots. The idea of synchronizing multiple coordinative robots to achieve a common goal is inspired by many examples found in nature. Our strategy is to control each robot to track along its desired trajectory while synchronizing its motion with the other robots to keep relative kinematics relationship as required by the coordination. To achieve this goal, we firstly pose the motion coordination problem as a synchronization control problem while defining the synchronization error based on the coordination requirement, and secondly we develop a synchronous controller that can guarantee both position and synchronization errors to approach zero asymptotically. Two case studies are conducted to demonstrate this synchronization approach. The first case study is to control formations of swarms of mobile robots to follow time-varying formations, with further extensions to various industrial applications such as coordination of multi-robot manipulators, multi-axis controls, and contouring error minimization of CNC machines. The second case study is to use a robotic cell manipulation system to transfer multiple biological cells in biomedical applications.

Furthermore, DARS 2014 had the Best Paper Award competition intended to recognize excellence among papers with substantial novelty and research contribution. The following papers were nominated in random order for the Best Paper Award by the Program Committee based on reviewer comments and scores.

- **Distributed Online Patrolling with Multi-Agent Teams of Sentinels and Searchers**
  Nicola Basilico[1], Timothy H. Chung[2], and Stefano Carpin[3]
  [1]University of Milan, Italy, [2]Naval Postgraduate School, USA, [3]University of California, USA
- **Human–Robot Collaborative Topological Exploration for Search and Rescue Applications**
  Vijay Govindarajan, Subhrajit Bhattacharya, and Vijay Kumar
  University of Pennsylvania, USA

- **Cooperative Mobile Robot Control Architecture for Lifting and Transportation of Any Shape Payload**
  B. Hichri[1], L. Adouane[2], J.-C. Fauroux[2], Y. Mezouar[2], and I. Doroftei[3]
  [1]Institut Pascal Clermont Ferrand, France, [2]Institut Pascal Clermont Ferrand, France, [3]Gheorghe Asachi Technical University of Iasi, Romania
- **A Repartitioning Algorithm to Guarantee Complete, Non-overlapping Planar Coverage with Multiple Robots**
  Kurt Hungerford[1], Prithviraj Dasgupta[1], and K.R. Guruprasad[2]
  [1]University of Nebraska, USA, [2]National Institute of Technology, India
- **A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration**
  Anshul Kanakia, John Klingner, and Nikolaus Correll
  University of Colorado, USA
- **Glider CT: Analysis and Experimental Validation**
  [1]Dongsik Chang, [2]Wencen Wu, and [1]Fumin Zhang
  [1]Georgia Institute of Technology, USA, [2]Rensselaer Polytechnic Institute, USA

The Best Paper Award went to Anshul Kanakia, John Klingner, and Nikolaus Correll for their paper "A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration." The winner was decided during the symposium by the Award Committee based on the technical merit and significance of the paper and quality of presentation. Listed below are the Program Committee and Award Committee members.

Program Committee

Asia/Oceania, Chair, Jun Ota (The University of Tokyo, Japan)
Marcelo H. Ang Jr. (National University of Singapore, Singapore)
Han-Lim Choi (KAIST, Korea)
Xavier Defago (JAIST, Japan)
Robert Fitch (University of Sydney, Australia)
Norihiro Hagita (ATR, Japan)
Sang Hoon Ji (KITECH, Korea)
Dongjun Lee (Seoul National University, Korea)
Geunho Lee (University of Miyazaki, Japan)
Azman Osman Lim (JAIST, Japan)
Makoto Mizukawa (Shibaura Institute of Technology, Japan)
Hyun Myung (KAIST, Korea)
Anton Satria Prabuwono (UKM, Malaysia)
Kwee-Bo Sim (Chung-Ang University, Korea)
Chieh-Chih Wang (National Taiwan University, Taiwan)

North America: Chair, Timothy H. Chung (Naval Postgraduate School, USA)
Nora Ayanian (University of Southern California, USA)
Spring Berman (Arizona State University, USA)
Sourabh Bhattacharya (Iowa State University, USA)
Zack Butler (Rochester Institute of Technology, USA)
Stefano Carpin (University of California, Merced, USA)
Nikolaus Correll (University of Colorado, Boulder, USA)
Karthik Dantu (University at Buffalo, State University of New York, USA)
Rafael Fierro (University of New Mexico, USA)
Emilio Frazzoli (Massachusetts Institute of Technology, USA)
Eric Frew (University of Colorado, Boulder, USA)
Maria Gini (University of Minnesota, USA)
Geoffrey Hollinger (Oregon State University, USA)
Ayanna Howard (Georgia Institute of Technology, USA)
Ani Hsieh (Drexel University, USA)
Kiju Lee (Case Western Reserve University, USA)
James McLurkin (Rice University, USA)
Nathan Michael (Carnegie Mellon University, USA)
Dejan Milutinovic (University of California, Santa Cruz, USA)
Michael Novitzky (Georgia Institute of Technology, USA)
Daniel Pack (University of Texas at San Antonio, USA)
Ioannis Rekleitis (McGill University, Canada)
Alessandro Renzaglia (University of Minnesota, USA)
Brian Sadler (US Army Research Laboratory, USA)
Ketan Savla (University of Southern California, USA)
Mac Schwager (Boston University, USA)
Wei-Min Shen (University of Southern California, USA)
Gabe Sibley (George Washington University, USA)
Stephen Smith (University of Waterloo, Canada)
Don Sofge (Naval Research Laboratory, USA)
Manuela Veloso (Carnegie Mellon University, USA)
Fumin Zhang (Georgia Institute of Technology, USA)

Europe, Chair, Fulvio Mastrogiovanni (University of Genoa, Italy)
Rachid Alami (LAAS/CNRS, France)
Francesco Amigoni (Politecnico di Milano, Italy)
Torbjorn Dahl (Plymouth University, UK)
Marco Dorigo (Université Libre de Bruxelles, Belgium)
Alessandro Farinelli (University of Verona, Italy)
Luca Maria Gambardella (IDSIA, Lugano, Switzerland)
Paolo Robuffo Giordano (IRISA/ INRIA Rennes, France)
Roderich Gross (University of Sheffield, UK)
Sandra Hirche (TUM, Munich, Germany)
Pedro U. Lima (Istituto Superior Tecnico, Lisbon, Portugal)
Lino Marques (University of Coimbra, Portugal)

Philippe Martinet (Ecole Centrale de Nantes, France)
Francesco Mondada (EPFL, Lausanne, Switzerland)
Daniele Nardi (Università La Sapienza, Rome, Italy)
Paolo Remagnino (Kingston University London, UK)
Antonio Sgorbissa (University of Genova, Italy)

Award Committee
Han-Lim Choi (KAIST, Korea), Chair
Marcelo H. Ang Jr. (National University of Singapore, Singapore)
Roderich Gross (University of Sheffield, UK)
Dongjun Lee (Seoul National University, Korea)
Hyun Myung (KAIST, Korea)
Mac Schwager (Boston University, USA)

We would like to offer our sincere thanks to the Organizing Committee members (Chair, Dr. Young-Jo Cho, ETRI, Korea) for their hard work and outstanding contributions, and the Steering Committee members (Chair, Prof. Hajime Asama, The University of Tokyo, Japan) for their helpful guidance and support. Special thanks are extended to the Program Co-chairs (Prof. Jun Ota, The University of Tokyo, Japan, Prof. Fulvio Mastrogiovanni, the University of Genoa, Italy, and Prof. Timothy H. Chung, the Naval Postgraduate School, USA) for their time and effort in attracting and recruiting qualified Program Committee members and collecting high-quality papers. We would also like to express our deep appreciation to the Program Committee and Award Committee members for their hard work and dedication. They all devotedly struggled to shape and maintain the highest quality levels of the final program within a very tight time frame. Last but not least, we would further like to express our heartfelt thanks and appreciation to all the participants for their active engagement in the symposium program and all the contributing authors in this volume.

Both academics and practitioners are invited to enjoy the very essence of DARS 2014, full of innovative ideas and practical strategies for implementation!

Nak-Young Chong
Young-Jo Cho

Contents

Part I
Collaborative Exploration, Localization, and Mapping

Distributed Online Patrolling with Multi-agent Teams of Sentinels and Searchers

Nicola Basilico, Timothy H. Chung and Stefano Carpin

Abstract We consider the problem of patrolling an assigned area using a team of heterogeneous robots consisting of sentinels and searchers in the presence of stochastic arrivals of attacks. Sentinels and searchers operate using a different sensor model featuring a tradeoff between accuracy and the sensed area. Using an approach based on queuing theory, we derive an accurate analytic characterization of the patrolling performance that can be used to predict the behavior of a given configuration or inform the composition of a team in order to meet a desired target performance. Extensive simulation results corroborate our theoretical findings.

Keywords Surveillance · Variable resolution search · Cooperative robots

1 Introduction

Among the many uses envisioned for teams of coordinated autonomous robots, tasks related to intelligence, surveillance and reconnaissance (ISR) continue to be at the forefront of research in distributed robotics. Teams of robots can implement search and patrolling strategies that complement and enhance human performance while reducing costs, increasing resilience, and decreasing operational risks for humans. Recent developments in the area of unmanned aerial vehicles (UAVs) have added momentum to this very active research area, in particular with the development of vertical take-off and landing vehicles, such as quadrotor UAVs [10].

N. Basilico
University of Milan, Milano, Italy
e-mail: nicola.basilico@unimi.it

T.H. Chung
Naval Postgraduate School, Monterey, CA, USA
e-mail: thchung@nps.edu

S. Carpin (✉)
University of California, Merced, CA, USA
e-mail: scarpin@ucmerced.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_1

In the recent past we have studied the problem of robotic search and patrolling using a single quadrotor UAV [1, 5]. Our initial modeling efforts and theoretical findings were corroborated with extensive field experiments demonstrating the validity of our assumptions [4]. A characteristic aspect of this class of vehicles is that their sensing resolution can be adjusted on the fly byKeeping this in mind varying their elevation—a fact already evidenced in [14]. Therefore, when planning how to allocate the search effort in space and time, one should also explicitly consider the variable sensor accuracy, defined here as detection probabilities. In fact, sensors and sensor processing algorithms have preferred operating conditions and one should strive to operate in those regions, when possible. Needless to say, operating in a regime offering the highest accuracy often comes at the cost of reducing the sensing area, thus creating opposing objectives. Our former works in this area have exactly explored this tradeoff in the single agent case.

In this paper we extend our existing work by considering how teams of heterogeneous robots can jointly patrol an assigned area. Our setup consists of two classes of robots, called *sentinels* and *searchers*. Sentinels and searchers operate at different elevations, and their sensors are then subject to different performances. The role of sentinels is to detect intrusions[1] and to then alert and dispatch searchers for their removal. Sentinels are stationary and capable of detecting intruders within large areas, whereas searchers are mobile and capable of removing the intruders, but their sensing area is much smaller. Both sentinels and searchers are equipped with faulty sensors incurring false alarms and missed detections. We model the problem using an approach based on queueing theory and we characterize the steady state behaviors of the queues using parameters characterizing the agents' sensors as well as the search strategy implemented by the searchers once dispatched. The derived model provides the basis for addressing various design and analysis questions. For example, we can anticipate the performance of a given composition and configuration of sentinels and searchers when contrasting different temporal and spatial stochastic profiles of intruder attacks. Alternatively, we can determine the optimal size and make-up of a team of sentinels and searchers in order to match a desired performance. Our approach is distributed in the sense that all processing is local to the agents and no information exchange is required. The only communication within the system is from the sentinels to the searchers, i.e., sentinels dispatch searchers when an intrusion is detected but sentinels do not communicate with each other, nor do searchers, respectively. By reducing the amount of exchanged communication and not having a centralized computational center, the resiliency of the system to individual failures increases—a key tenet of distributed robotic systems.

The rest of the paper is organized as follows. Selected related works are discussed in Sect. 2. The formal definition of the problem is given in Sect. 3. The formalization based on queueing theory is given in Sect. 4 and simulations substantiating our findings are presented in Sect. 5. Finally, conclusions and future works are given in Sect. 6.

[1] Throughout this paper we use terms like *intrusion*, *attack* and the like that come from the security games literature. Clearly these events encompass a broader scope and may be related to phenomena not necessarily triggered by an antagonistic opponent.

2 Related Work

Algorithmic models for addressing the proposed patrol problem and its variants have been explored extensively in various communities, including robotics, operations research, and industrial engineering, with operational relevance and significant impact in application areas such as law enforcement, perimeter security, and shipping logistics. Of closest relevance to this paper are formulations of the dynamic vehicle routing problem in relation to algorithmic queuing theory, such as in [3, 8], in which events requiring servicing appear in the environment stochastically, such as random arrivals of intruders in a protected area, requiring one or more agents to prioritize and visit these locations in an online manner. Alternate formulations consider patrol sequences under different assumptions for intruder arrivals, such as cases where intrusion sites are determined according to known probability distributions or by assuming adversarial intruders requiring game-theoretic design of patrols [9]. Commonly used objectives in such patrol problem formulations include minimizing the average or worst case revisit rate to return to a given location, which has correspondence to measures of service rates and wait time in queuing theory models [7, 12]. Other metrics, such as maximized area coverage for sensor deployments [2, 13], enable decentralized control laws to govern persistent surveillance of areas. However, these previous models do not incorporate the possibility of imperfect detections of the events, for which Bayesian methods found in probabilistic search theory [5, 6] provide key insights.

The main theoretical and algorithmic contributions of the proposed work address the challenge of persistently surveilling an area with distributed probabilistic sensors, both fixed and mobile, that are prone to false positive and false negative detections. In addition, this paper highlights insights into the tradeoff in using multi-scale representations of the environment with varying numbers and compositions of such heterogeneous sensors.

3 Problem Definition

We consider the problem of patrolling a planar region using a team of multiple UAVs. We adopt a discretized representation of the environment, namely a regular grid $\mathscr{G}$ composed by k equally sized square cells. Any cell c can be the target of a malicious activity referred as *attack*. Attacks can be ongoing in one or multiple cells at any given time and only searchers can remove them by performing a *clear* action. A *loss function* $l : \mathscr{G} \to \mathbb{R}_{\geq 0}$ assigns to each cell c a value $l(c)$ which is the cost incurred per unit time while an attack is taking place at cell c.

Given this general background, we define a metric to evaluate the performance of any team of agents independently from their number and their coordination mechanism. Similarly to the metric we introduced in [1], let $a(c, t)$ be a function describing the spatio-temporal realization of attacks, where $a(c, t) = 1$ if at time t an attack is

present in cell c, and $a(c,t) = 0$ otherwise. Without loss of generality, we assume that a patrolling mission starts at time $t = 0$ and ends at time T. We then define our performance metric as:

$$\rho(T) = \sum_{c \in \mathcal{G}} l(c) \int_0^T a(c,t)dt \tag{1}$$

Equation 1 is the sum of k integrals, each measuring the time every cell is under attack (scaled by the loss attributed to the cell). Differently from [1], here we consider continuous time to ease our subsequent theoretical analysis based on queueing theory.

The heterogeneous patrolling team consists of $N = M + R$ agents, with $1 \leq M \leq |\mathcal{G}|$ and $R \geq M$. M agents are of type *sentinel* while R are of type *searcher*. Sentinels are stationary agents which repeatedly scan large portions of the environment for the presence of an attack in that region. When a sentinel observes a positive reading, it dispatches a searcher. Searchers are instead moving agents capable of conducting fine-grained, find–and–clear tasks over some area. Searchers try to localize and clear attacks within the region assigned to the sentinel that dispatched them. In pursuing such task, they will follow some search strategy and will be subject to a finite temporal budget limit. Due to the limited temporal budget and to the use of faulty sensors resulting in missed detections, searchers may fail to detect and remove an intruder present in their assigned area.

The stochastic process of attacks. We consider a situation where the environment is constantly under the threat of attacks which can randomly occur at any time and at any place. We adopt a common assumption from patrolling literature (see, for example, [3]) according to which arrival times for attacks obey a Poisson distribution while their spatial location is determined according to some discrete probability distribution over $\mathcal{G}$. More formally, the inter-arrival time in the whole environment is modeled by an exponential variable with parameter λ while the specific cell c is chosen with a probability proportional to the value $l(c)$, i.e., once an attack arrived in the environment, the probability that it will be located at cell c is

$$\Pr[c] = \frac{l(c)}{\sum_{m \in \mathcal{G}} l(m)}. \tag{2}$$

Once started in a cell c, an attack persists until it is eventually cleared. Note that, based on this model, the same cell may suffer from multiple concurrent attacks.

Defending the environment with sentinels and searchers. Each sentinel i is stationed at a fixed location and is tasked with monitoring a sub-portion of the environment $\mathcal{G}_i \subset \mathcal{G}$. Different assumptions can be made on how sub-portions are defined and where sentinels are positioned. For example, if a Voronoi partition is used, the sentinel could occupy the generator points associated with each partition [2]. Consistent with our sensor model, we adopt a representation based on probabilistic quadtrees [4]. Each sentinel guards a rectangular area $\mathcal{G}_i$, and all $\mathcal{G}_i$s constitute a partition of $\mathcal{G}$ (see Fig. 2 for a visual representation). All areas assigned to the

sentinels are equally sized and sentinels are therefore all positioned at the same elevation. With each sensing action, sentinel i obtains a binary reading which, if positive, is interpreted as evidence that at least one attack is currently present in $\mathscr{G}_i$. No additional information is provided for the location of the attack, that is, uniform spatial uncertainty over the cells of $\mathscr{G}_i$ is assumed. With probability α, a sentinel receives a positive reading even if no attack is present in its region (false positive), and with probability β, a negative reading occurs when at least one attack is present (missed detection). Such false positives and missed detection rates depend on the sensor resolution, e.g., defined by the altitude, at which sentinels are located (see [4]). Each sentinel inspects its assigned area for the presence of attacks on a periodic basis every Δ time units.

As soon as a sentinel receives a positive detection (whether true or false), a searcher is dispatched over $\mathscr{G}_i$. A searcher's objective is to find and clear ongoing attacks in that area. To this end, it searches the area to determine which cells within $\mathscr{G}_i$ are under attack. Once a cell c is believed to be under attack above some level of confidence, a clearing action is undertaken. We assume that when such action is performed, if an attack is indeed present, then it is always neutralized. (If more attacks are present, then only the one that has been residing there for the longest time is cleared in a FIFO fashion).

The execution of this task poses the problem of using a patrolling strategy with which a searcher can be driven in the decisions about where to sense next and when to perform clearing actions trying, at the same time, to locally minimize the performance metric. In introducing our two–type based architecture, we opt for searchers driven by *deterministic* strategies. Such strategies are defined as cyclically repeated paths that scan every cell on a periodic basis. Examples of such strategies are the sweep and the lawn mower patterns [6]. In fact, from a practical perspective these strategies are nowadays still the most widely used in the field. The reason for this restriction to deterministic strategies comes from their relatively simple characterization under statistical terms. This allows us to provide a neat theoretical analysis of our two–type approach, without the cumbersome technicalities that more complex patrolling strategies would have introduced. Such investigations belong outside the scope of this paper and will be the subject of our future research on this problem.

We assume that each searcher is given a time budget B. As soon as such budget is completely consumed, the searcher ends its patrolling mission and returns to the base station. Just like sentinels, each searcher s is equipped with faulty detection sensors whose false positive and missed detection rates are denoted with α_s and β_s, respectively.

4 Theoretical Analysis

Our eventual objective is to evaluate the loss value defined in Eq. 1 as a function of the various parameters characterizing the system. Inspired by [3], in this section we answer this question relying on queueing theory. The relevant parameters are:

- α, β: false positive/missed detection rates by each of the sentinels;
- Δ: interval between two successive scans by a sentinel;
- A: probability that an attack occurs within the area assigned to the sentinel guarding cell c during an interval of time Δ;
- $N = 1 - A$: probability that no attack occurs;
- α_s, β_s: false positive/missed detection rates of each of the searchers (in general different than values for the sentinels).

To evaluate the loss, we associate a queue to each cell c and we determine its steady state behavior. This is simpler than modeling all attacks occurring in $\mathscr{G}$ with a single queue. Once the steady behavior of each queue is determined, the overall loss can be evaluated simply adding up the loss accrued in each individual cell. Let Q_c be the queue associated with cell c. Using Kendall's notation, Q_c can be modeled as a $M/G/1$ queue. Note that in general each of the queues is characterized by a different set of parameters. In particular, they will depend on the value $l(c)$. The assumption that the service time is generic (G) stems from the choice of search strategy, i.e., the sweep pattern. Little's theorem [11] states that the expected number of elements L_c in the Q_c is

$$L_c = \lambda_c W_c,$$

where λ_c is the arrival intensity (number of arrivals per unit time) and W_c is the expected time spent in the queue. Note that this theorem does not rely on Markovian assumptions on the processes, but only on the ergodicity of said stochastic processes and is therefore applicable also for $M/G/1$ queues. Once we know L_c for each cell, then through Eq. 1 we can compute the expected aggregate loss. In the following we construct λ_c and W_c for the generic queue Q_c.

Interarrival time. The process governing the intruders' spatial and temporal behavior is described in Sect. 3. The interarrival time between two intruders entering the patrolled area is modeled by an exponential variable with parameter λ, such that the expected interarrival time in the patrolled area is $1/\lambda$. Upon an intruder's arrival, it determines the specific cell c to attack according to the mass distribution defined by Eq. 2. The number of attacks necessary before c is attacked is then modeled by a geometric variable with parameter $p(c)$ and its expectation is $1/p(c)$. Thus, the expected interarrival time for a specific cell c incorporates both temporal and spatial components, given by

$$T_c = \frac{1}{\lambda}\frac{1}{p(c)} \tag{3}$$

and the arrival intensity for cell Q_c is then $\lambda_c = \lambda p(c)$.

Service time. We next need to determine W_c, i.e., the expected time spent in Q_c by an intruder. Figure 1 depicts the most general case that helps in understanding the structure of the random variable w_c, modeling the time spent by an intruder before it is removed.

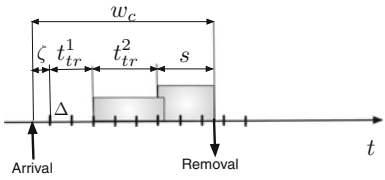


Fig. 1 Elements contributing to the time w_c between the arrival and removal of an intruder

The sentinel queries its sensor at a fixed frequency and once a searcher is dispatched to the area, it may or may not find all intruders. w_c is then the sum of various components. The first component, ζ, is the time elapsed between when the intruder arrives in cell c (and then enters Q_c) and the first time the sentinel scans the area including c. In general it may take more than one scan before a searcher is dispatched. This time is given by the variable t_{tr} (time to trigger) and by construction, a multiple of Δ. After a searcher is dispatched, it will not necessarily find the intruder, so in general multiple successive, independent searchers have to be dispatched. Then, t^i_{tr} is the time to trigger the dispatch of the ith searcher. Once the successful searcher is dispatched, it spends time s before it finds the intruder. Therefore,

$$w_c = \zeta + \sum_{i=1}^{n_s} t^i_{tr} + s$$

where n_s, the number of dispatched searchers, is also a random variable. As it will be explained later on, the various t^i_{tr} are all independent but not all equally distributed. In particular, t^1_{tr} has a distribution different from the following ones, whereas all the t^j_{tr} with $j \geq 2$ are i.i.d. Keeping this in mind, we can then write

$$W_c = \mathbb{E}[w_c] = \mathbb{E}[\zeta] + \mathbb{E}[t^1_{tr}] + (\mathbb{E}[n_s] - 1)\mathbb{E}[t_{tr}] + \mathbb{E}[s]. \tag{4}$$

Let us start with computing $N_s = \mathbb{E}[n_s]$. Each searcher follows a deterministic search strategy with a finite time budget. During this search each cell is inspected the same number of times, say m. The missed detection error rate is β_s, so a searcher fails to find the intruder located in c with probability β_s^m, and finds it with probability $1 - \beta_s^m$. The number of searchers, n_s, needed to detect the intruder is then modeled by a geometric variable with parameter $1 - \beta_s^m$ and its expectation is $\mathbb{E}[n_s] = \frac{1}{1-\beta_s^m}$.

Next, we determine $S = \mathbb{E}[s]$ conditioned on the event that the searcher finds the intruder. Given that the searcher follows a predetermined path unrelated to $l(c)$, assuming that the search time budget is B, then $S = B/2$ because the intruder could be located with equal probability in any of the sequentially scanned cells.

To determine $Z = \mathbb{E}[\zeta]$, it is useful to recall that the interarrival time of Q_c and then of the intrusions to cell c is modeled by an exponential variable of parameter $\lambda_c = \lambda p(c)$. Due to the memoryless property of exponential random variables and its basic properties, it follows that $\zeta = \Delta - y$, where y is an exponential random variable of parameter λ_c conditioned on the event $y \leq \Delta$. Through algebraic manipulation and applying the definition of expectation one obtains

$$\mathbb{E}[\zeta] = \mathbb{E}[\Delta - y] = \Delta - \mathbb{E}[y] = \Delta - \frac{1 - e^{-\lambda_c \Delta} - \lambda_c \Delta e^{-\lambda_c \Delta}}{\lambda_c (1 - e^{-\lambda_c \Delta})}$$

Finally, it is necessary to compute $\mathbb{E}[t_{tr}^1]$ and $\mathbb{E}[t_{tr}^j]$ with $j > 1$. Recalling that t_{tr}^i is an integer multiple of Δ, i.e., $t_{tr}^i = K\Delta$, it is then sufficient to compute the mass distribution of the multiplicative factor K for the two different cases. K is the number of times the sentinel has to sense before a searcher is dispatched. It is useful to recall that t_{tr}^1 models the time to trigger the dispatch of the first searcher conditioned on the fact that one intruder entered the area assigned to the sentinel (see Fig. 1). Its mass distribution is

$$\Pr[K = k] = \begin{cases} (1 - \beta) & k = 1 \\ \beta(A\beta + N(1 - \alpha))^{k-2}(N\alpha + A(1 - \beta)) & k \geq 2 \end{cases} \tag{5}$$

The rationale behind the formula is the following. $K = 1$ if the intruder generates a detection by the sentinel the first time the sentinel senses the area. This is by definition $1 - \beta$. Otherwise, conditioned on the fact that an intruder entered cell c, the first searcher will be triggered after $k \geq 2$ scans as a consequence of the simultaneous occurrence of the following independent events:

- the intruder is not detected, which has probability β;
- for $k - 2$ steps there was not detection. Since each step is independent from each other, we can just raise to the power of $k - 2$ the probability that no detection occurred in one step. This event is either due to an attack going undetected, whose probability is $A\beta$ or a non-attack not generating a false positive (probability $N(1 - \alpha)$). Note that these two events are mutually exclusive (either an attack happens or it does not), so we can just add the probabilities together.
- at the last step a detection happens. This is either due to a non-attack generating a false positive (probability $N\alpha$) or an attack being detected (probability $A(1 - \beta)$).

We seek an expression for the mass distribution for t_{tr}^j, i.e., the time to trigger the jth searcher ($j > 1$) conditioned on the fact that the first searcher has already been dispatched. This variable is a geometric random variable, and its distribution is then:

$$\Pr[K = k] = (N(1 - \alpha) + A\beta)^{k-1}(A(1 - \beta) + N\alpha).$$

The rationale to derive this formula is similar to the one for t_{tr}^1 and one should also notice that it is indeed a geometric variable because $N(1 - \alpha) + A\beta + A(1 - \beta) + N\alpha = 1$. To complete the computation of Eq. 4 we need to compute $\mathbb{E}[t_{tr}^1]$ and $\mathbb{E}[t_{tr}]$. Skipping the algebraic details in the interest of space, we just give the results, i.e.,

$$\mathbb{E}[t_{tr}^1] = 1 - \beta + \frac{\beta}{1 - N\alpha - A(1 - \beta)} \qquad \mathbb{E}[t_{tr}] = \frac{1}{1 - A(1 - \beta) + N\alpha}.$$

We conclude this section noting that A and N can be easily determined from knowledge of the set of cells covered by the sentinel guarding cell c.

5 Simulations

In this section, we provide experimental analysis for empirical assessment of some of the properties of the proposed two-type approach. We analyze performance in terms of accrued loss (per Eq. (1)), required costs in terms of number of deployed sentinels and frequency of sampling for each of them, and we look at how the system responds under different loads expressed by variable attack arrival rates.

Our basic experimental setting builds on top of the in-the-field validation conducted in [4] with the aim of maintaining relevance to realistic deployments of UAVs. The grid $\mathscr{G}$ consists of 16×16 cells and two different loss functions are considered: a simple uniform loss (UNI) that assigns equal loss to every cell and a bimodal one (BI) depicted in Fig. 2a. The arrival rate for attacks is $\lambda = 1/95$.

We consider three different groups of sentinels of cardinality 1, 4, and 16 uniformly deployed in the environment (see Fig. 2b, c). That is, if h sentinels are present, then their equally sized assigned areas $\mathscr{G}_i$ constitute a partition of $\mathscr{G}$. The sampling period for each sentinel is given by $\Delta = L/4$, where L is the time a searcher would require to scan and clear every cell of its area by following some deterministic strategy. Error rates are chosen as a function of the altitude and, to account for the fact they are tailored for constant altitude, we scaled by a $\frac{1}{2}$ factor, that is $\alpha^1 = 0.43/2$, $\beta^1 = 0.38/2, \alpha^4 = 0.36/2, \beta^4 = 0.27/2, \alpha^{16} = 0.35/2, \beta^{16} = 0.37/2$ where α^h and β^h refer to the error rates when having a deployment of h sentinels. These error rates, as well as those for the searchers given in the following, were determined from extensive live-fly experiments presented in [4].

Searchers conduct a deterministic sweep pattern, sequentially scanning every cell per unit time on the sub-grid associated to that area. False positives and missed detections are chosen according to their altitude value (the lowest in a quadtree built over a 16×16 grid) as $\alpha_s = 0.09$ and $\beta_s = 0.05$. We assume that flying from a cell to an adjacent one, scanning that cell, and performing a clear action on the cell each take a single time unit. As a consequence, scanning and clearing every cell of a sub-grid $\mathscr{G}'$ takes $L = 3|\mathscr{G}'|$ time units. We also define the temporal budget of each searcher w.r.t. this quantity as $B = mL$, where m is an integer value. In the results presented here, we fix $m = 2$, that is, once dispatched, each searcher must always perform at least two whole sweeps of the assigned area. Finally, we assume

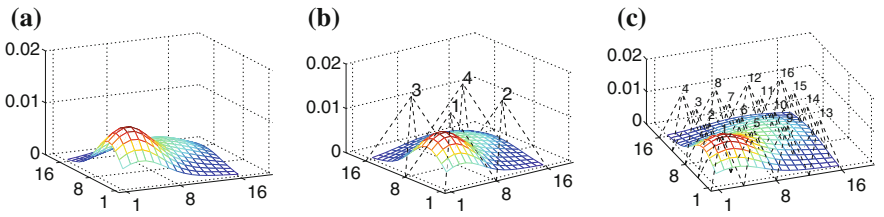


Fig. 2 Bimodal loss function and deployments of multiple sentinels. **a** Bimodal loss. **b** Deployment of 4 sentinels. **c** Deployment of 16 sentinels

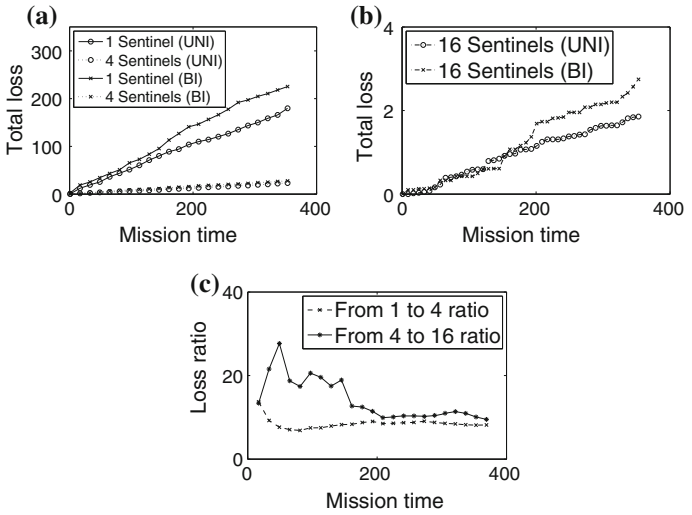


Fig. 3 Total loss accrued during missions for varying number of sentinels for different loss functions

to have an unbounded number of searchers, namely every dispatch is immediately executed. Studying the situation where the number of searchers is bounded is left for future work, although as evidenced in this section, the number of concurrently active searchers remains limited.

Figure 3 reports average results obtained for an experimental design of 20 random missions. (Each run corresponds to a different realization of the attacks arrival process.) Graph 3a, b show how the average accrued loss evolves as the mission unfolds and as sentinels sense their areas every scan period k.

By inspecting these graphs, we can empirically assess the extent of two expected trends in the actual performance achieved by the different teams. The first observation is that having more sentinels leads to a smaller loss whose reduction is nearly optimal when employing 16 sentinels. One interesting feature can be observed in Fig. 3c where the ratios between single-sentinel and 4-sentinel loss as well as between 4-sentinels and 16-sentinels are depicted (bimodal loss is considered here). The first thing we notice is that even if we increase our resources by a factor of 4, we observe (mostly at every mission time) gains of much higher order ($\geq$10). The reason is that, besides merely having more sentinels we are also introducing two improvement factors, which indirectly come by construction of our framework: (1) not only are sentinels greater in number but also are each more accurate in sensing, since they operate at lower altitudes; (2) the more sentinels are employed, the more effectively the environment is split for parallel patrolling missions (for any loss function). This second factor contributes to the other observed trend, that is, passing from 4 to 16 sentinels is never worse than increasing from 1 to 4. Indeed, when deploying 16 sentinel we get a critical split of an highly targeted sub-area of the environment (recall Fig. 2).

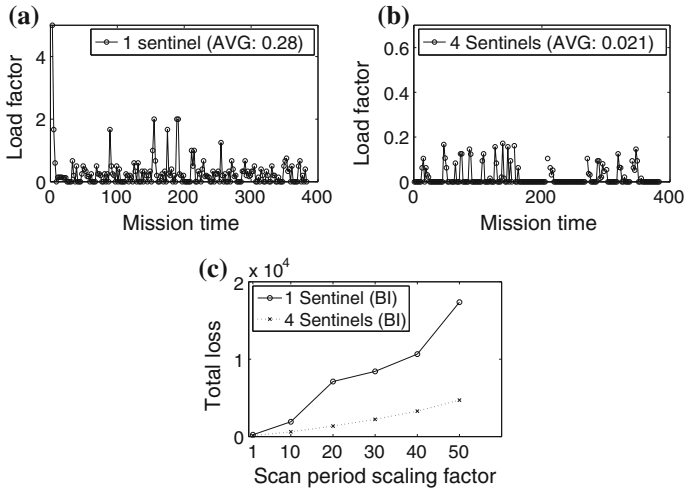


Fig. 4 Load factors and accrued loss with different scan periods durations. **a** 1 Sentinel. **b** 4 Sentinel. **c** Loss w.r.t. scan period

Moreover, from Graph 3a, b we can see how a bimodal loss function results in poorer performance, showing the disadvantage of adopting a uniform spatial deployment over a non-uniform loss distribution.

An interesting operational metric is given by the load factor of each sentinel, defined as the ratio between the total number of attacks still present in the environment over the number of searchers that have been dispatched by that sentinel and did not use up their respective time budgets. Figure 4a, b compare average factors for the 1-sentinel and 4-sentinels cases as the mission evolves (the curve in Fig. 4b depicts the average load factor over the four sentinels). The 1-sentinel case reported an overall average load of 28 %, whereas, as it can be seen, different mission times experienced an overload condition (load factor greater than one) with attacks outnumbering searchers. Such situation is not observed when employing four sentinels, and the overall average factors for each sentinel resulted to be remarkably lower. Such results experimentally highlight the improvements obtained from the partitioning of the search area, w.r.t. a metric which is independent from the loss function, i.e., the importance level assigned to every cell in the grid.

The number of sentinels constitutes the primary measure of cost in our setting. Another important cost factor is given by the number of employed searchers or, equivalently, the number of dispatches. Given the assumption of an unbounded R, we can control such cost via the sentinels scanning period Δ, with the obvious expectation that the more frequently sentinels scan the more dispatches they will likely issue. Figure 4c shows how reducing costs of this type can introduce a decrease in performance. Starting from our reference value of Δ, we scale it by increasing integer factors and we measure the total loss accrued at the end of the mission. As can be seen, the 1-sentinel case is where longer scan periods are more critical.

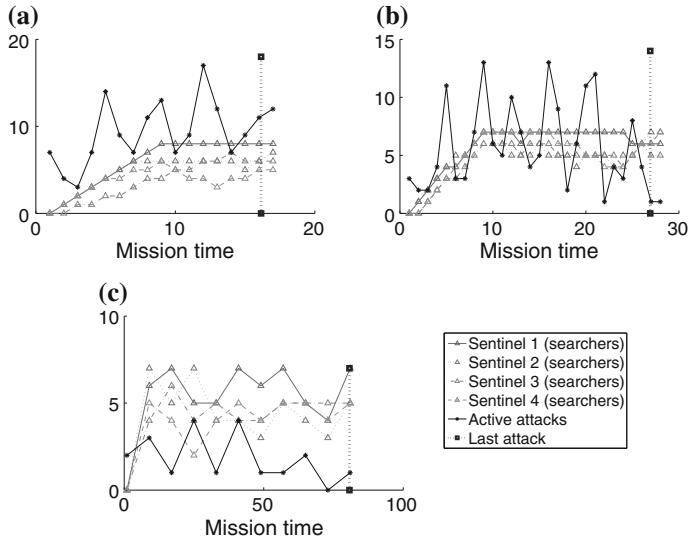


Fig. 5 Number of active searchers and attacks with different arrival rates. **a** 100λ. **b** 60λ. **c** 20λ

On the contrary, situations with multiple sentinels (e.g., the case of four sentinels included in the graph) seem to be more robust with a relatively graceful degradation in performance.

For further experimental validation, we assess how the system responds to increasing attack arrival rates by showing in Fig. 5 how the number of active searchers and attacks vary during the mission under arrival rates obtained by scaling our reference λ. The observed trend is that for very high arrival rates, the number of attacks almost always exceeds the number of active searchers for each sentinel. (Note that the number of attacks is the per-sentinel average where an attack is associated to a sentinel if it occupies a cell in that sentinel's area.)

In our final experiment we assess the sensitivity of our model to the parameters characterizing the stochastic model of attacks. In particular, we focus on the interarrival times. Our analysis stands on the assumption that these random variables are i.i.d and follow an exponential distribution with known parameter λ. In our last test we change this distribution with a different one having the same expectation. This choice is motivated from practical considerations. When building a model of the opponent through repeated observations, experimentally observing the expected interarrival time is the simplest first step, but there are evidently multiple distributions that can fit the data. In this experiment, we select a uniform distribution. Figure 6 plots the difference between the performance of the system under two different scenarios. In the first case interarrival times are distributed according to an exponential distribution and then match the model we used in deriving our analysis. In the second case interarrival times are uniformly distributed, but now incorrectly modeled. As we did for Fig. 3a, we vary the number of sentinels (one or four) and consider two different

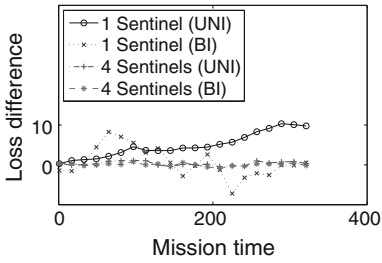


Fig. 6 Difference in accrued loss when interarrival times arrivals are exponentially distributed or uniformly distributed

loss models (unimodal or bimodal), thus obtaining four different curves. The figure shows that when considering four sentinels, differences in performance are negligible. When a single sentinel is considered, a difference, albeit limited, is observed. To put the magnitude of the difference into perspective, the reader is referred to Fig. 3a for absolute loss values. Given that in general one will use multiple sentinels, these findings tend to indicate that the model is robust to identification errors for interarrival times.

6 Conclusions

In this paper we have studied a patrolling problem using two classes of agents, namely sentinels and searchers. The setup is inspired from our recent work in a single-agent setting and our model is driven by experimental data collected through extensive live-fly experiments. Using analytic formulations founded on queueing theory, it is possible to determine how the system behaves asymptotically in response to different stochastic models of arrivals. Studies in simulation show how explicitly modeling a variable resolution sensor leads to gains outweighing the potential penalty of increasing the number of allocated sentinels.

Future work include extensions explicitly handling deconfliction and coordination among searchers, as well as deploying sentinels with overlapping regions for increased robustness and performance. Additional research addresses further theoretical analysis of the impact of constrained resources (e.g., number of searchers available to sentinels), with relevance to realistic deployments. Finally, building upon the analysis we developed, we will consider how to non-uniformly allocate sentinels in the environment in order to minimize the given performance metrics. This includes positioning more sentinels to cover areas with higher loss values, as well as varying their elevation to operate in regimes with lower error rates where needed.

References

1. Basilico, N., Carpin, S.: Online patrolling using hierarchical spatial representations. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2163–2169 (2012)
2. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Princeton University Press, Princeton (2009)
3. Bullo, F., Frazzoli, E., Pavone, M., Savla, K., Smith, S.L.: Dynamic vehicle routing for robotic systems. Proc. IEEE **99**(9), 1482–1504 (2011)
4. Carpin, S., Basilico, N., Burch, D., Chung, T.H., Kölsch, M.: Variable resolution search with quadrotors: theory and practice. J. Field Robot. **30**(5), 685–701 (2013)
5. Chung, T.H., Carpin, S.: Multiscale search using probabilistic quadtrees. In: Proceeding of the IEEE International Conference on Robotics and Automation, pp. 2546–2543 (2011)
6. Chung, T.H., Silvestrini, R.T.: Modeling and analysis of exhaustive probabilistic search. Naval Res. Logist. **61**(2), 164–178 (2014)
7. Enright, J.J., Frazzoli, E.: Optimal foraging of renewable resources. In: American Control Conference (ACC), 2012. IEEE, pp. 683–690 (2012)
8. Huynh, V.A., Enright, J.J., Frazzoli, E.: Persistent patrol with limited-range on-board sensors. In: 2010 49th IEEE Conference on Decision and Control (CDC), pp. 7661–7668. IEEE (2010)
9. Lin, K.Y., Atkinson, M.P., Chung, T.H., Glazebrook, K.D.: A graph patrol problem with random attack times. Oper. Res. **61**(3), 694–710 (2013)
10. Mahoni, R., Kumar, V., Corke, P.: Multirotor aerial vehicles—modeling, estimation and control of quadrotor. IEEE Robot. Autom. Mag. **19**(3), 20–32 (2012)
11. Papoulis, A., Pillai, A.U.: Probability, Random Variables and Stochastic Processes. McGraw-Hill, New York, 2002
12. Pippin, C., Christensen, H., Weiss, L.: Performance based task assignment in multi-robot patrolling. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 70–76. ACM (2013)
13. Schwager, M., Julian, B.J., Angermann, M., Rus, D.: Eyes in the sky: decentralized control for the deployment of robotic camera networks. Proc. IEEE **99**(9), 1541–1561 (2011)
14. Waharte, S., Symington, A., Trigoni, N.: Probabilistic search with agile UAVs. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2840–2845 (2010)

Human-Robot Collaborative Topological Exploration for Search and Rescue Applications

Vijay Govindarajan, Subhrajit Bhattacharya and Vijay Kumar

Abstract We address the coordination between humans and robots in tasks that involve exploration and reconnaissance with applications to search and rescue. Specifically, we consider the problem of humans and robots cooperatively searching an indoor environment in a distributed manner where we assume that each robot is equipped with sensors that are able to locate targets of interest. Rather than have humans issue explicit commands to and guide robots, we allow humans to make decisions on their own and let the robots adapt to decisions taken by the human. The main contribution of this paper is a framework in which the robots in the team respond and adapt to the behavior of the human agents in the task of exploring and clearing an indoor environment. The central idea is the assignment of robots to homotopy classes that are complementary to the classes being pursued by human agents. By the virtue of the sparse topological representation of the agent trajectories, our algorithm lends itself naturally to a distributed implementation. The framework has three advantages: it (a) ensures that robots and humans pursue different homotopy classes; (b) requires very little communication between the humans and the robots; and (c) allows robots to adapt to human movement without having to model complex human decision-making behaviors. We demonstrate the effectiveness of the proposed algorithm through a distributed implementation on a ROS (Robot Operating System) platform.

Keywords Search and rescue · Path planning · Homotopy · Human-robot interaction

V. Govindarajan (✉)
Department of Electrical Engineering, University of Pennsylvania, Philadelphia, USA
e-mail: govvijay@seas.upenn.edu

S. Bhattacharya
Department of Mathematics, University of Pennsylvania, Philadelphia, USA
e-mail: subhrabh@math.upenn.edu

V. Kumar
Department of Mechanical Engineering and Applied Mechanics,
University of Pennsylvania, Philadelphia, USA
e-mail: kumar@seas.upenn.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_2

1 Introduction

We address the coordination between humans and robots in tasks that involve reconnaissance with applications to search and rescue. In these applications, robots may need to quickly and safely explore environments in collaboration with human counterparts. When confronted with two or more hallways, a human first responder may choose to explore one hallway, while his/her robot co-responder explores a different hallway. Similarly, in teams of multiple human and robot explorers, we want the exploration task to be naturally decomposed between the team members. At the same time, we want the human(s), who are better at interpreting the available information and at decision making, to decide what actions they want to take and let robots adapt accordingly.

We consider a setting where humans and robots are equipped with similar sensing, processing, and communication capabilities, so that robots and humans can be aware of each others' positions and robots can interpret human movements and intentions. The sensing capabilities of the agents are assumed to provide adequate range to detect anomalies (e.g., victims or intruders) in an environment. We assume that both humans and robots have access to blueprints of buildings and are thus aware of the major features in the environment. As a result, both humans and robots are able to localize themselves with respect to features in the buildings using onboard sensors such as laser scanners, cameras and IMUs, as well as GPS, if available. Finally, we assume that the human-worn computers are able to communicate with the robot co-workers and share their estimates of current location periodically.

Metric-based multi-robot coordinated exploration have been studied widely in the past [1–5]. Multi-robot coverage of environments are also fundamentally considered as metric problems relying inherently on a metric on the configuration space [6–10]. In addition, graph traversal approaches similar to the *traveling salesman* problem [11] have been explored in context of room-clearing [12] and pursuit-evasion problems [13]. Similar coverage problems can be formulated as a traversal on *Voronoi graph* or *topological map* [14, 15] of an environment.

However, in a problem setting as ours, it is likely that maps may not be perfect. Noise in the process dynamics and observations will induce errors in localization. Thus representations derived from metric information will require estimation techniques that yield estimates of states that are stochastic. Topological invariants such as homotopy, on the other hand, being robust towards environmental noise and measurement errors, are suitable for communication and coordination among the heterogeneous teams of humans and robots. Furthermore, our primary objective being quick information/intelligence gathering and clearing, it is not necessary that the agents visit every point in the environment (as done in graph traversal algorithms). Homotopy classes of trajectories form natural equivalence classes, within each of which the information available are similar. If two trajectories belong to the same homotopy class, then a single agent can perform the task of gathering the available information in that class, while diverting additional resources to other classes. While homotopy is directly related to visibility in most indoor environments

(e.g. consisting mostly of hallways and corridors), even in presence of non-convex features within the class (e.g. a room by the corridor), traversing the class is sufficient to gather intelligence and information from the adjacent features as those (e.g., by glancing through the doors of the rooms) and does not need dedicated agents for each of those features. Another advantage of using homotopy classes as the primary means of decision making in coordinated reconnaissance and clearing is the simplicity of its representation, and thus the ease and efficiency in communication. Choosing *complementary homotopy classes* by the robot agents is achieved naturally and efficiently, and such choices can be easily adapted to change in the human actions. Such algorithmic simplicity is absent in graph traversal approaches.

There has been some recent research on using topological techniques in exploration of environments [16]. In this paper our fundamental approach is topological as well. We exploit topological features in the environment, namely the different *homo-*

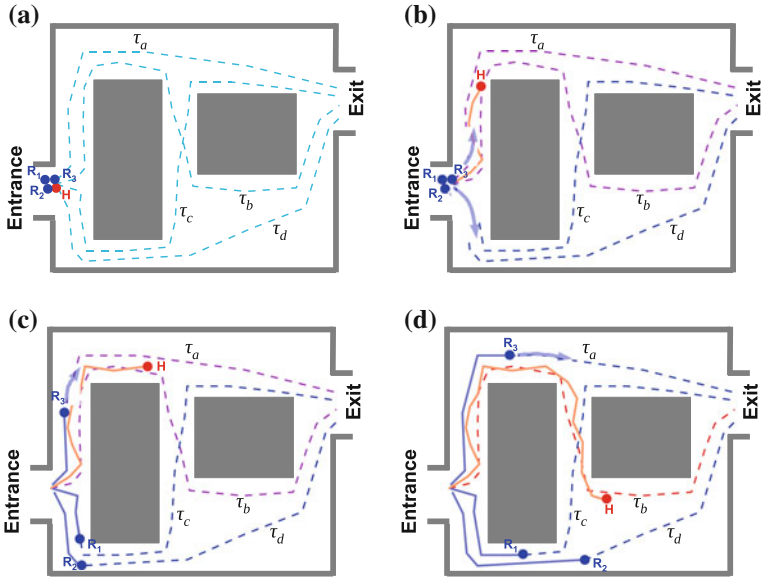


Fig. 1 A simple illustration of the idea behind the proposed algorithm involving a team of one human and three robots. Robots must respond to human action by choosing paths in homotopy classes complementary to those taken by humans to maximize exploration. The algorithm takes into account teams consisting of arbitrary number of humans and autonomous robots. In addition, robots can effectively adapt to erratic/unpredictable human behavior (not illustrated in this figure), where a human, after committing to a class, may turn back to choose a different homotopy class. **a** Four agents, three robots (R_1, R_2, R_3) and one human (H), enter an environment with 4 homotopy classes of paths ($\tau_a, \tau_b, \tau_c, \tau_d$) leading to the exit. The robots wait for the human to move first. **b** Based on human's initial trajectory (*solid curve*), the robots infer that H is taking the homotopy class of τ_a or τ_b. The homotopy classes of τ_c and τ_d are thus to be taken by robots. **c** R_3 tailgates the human (to pursue τ_a or τ_b—whichever not taken by H), while robots R_1 and R_2 commit to τ_c and τ_d. **d** H has committed to τ_b, and thus R_3 commits to homotopy class of τ_a

topy classes of trajectories, to guide our search and rescue missions. This topological reasoning is fundamental in deciding how the autonomous agents respond to human behaviors. Although we do use a metric on the space of trajectories in the workspace (*Hausdorff distance*), this is purely as an intermediate step towards classifying a human's trajectory into one or more of the homotopy classes.

Our algorithm design is inspired by the need to keep explicit human-robot communication (e.g., human commanding robots) at minimum in a time-critical mission such as search and rescue. The humans should have the freedom to choose actions based on their superior sensing ability (e.g., audio cues) and change actions without having to explicitly communicate intent to other agents. The robot agents should be able to adapt to the human actions and choose complementary tasks to maximize efficiency of coordinated survey and search. We illustrate the problem at hand using the scenario in Fig. 1 where there is one human and three autonomous robots. All the agents enter through a single entrance in the environment and need to clear the building and reach the exit. The figures illustrate how the robots take decisions and respond based on the human agent's actions. Our proposed algorithm is highly suited for a distributed implementation, requiring only limited inter-agent communication of coarse topological representation (h-signature) of their trajectories.

2 Background

In this section, we will review some preliminary definitions and algorithmic tools.

2.1 Homotopy Class of Trajectories

Suppose $W \subseteq \mathbb{R}^2$ is a simply-connected workspace for the agents with m counts of connected obstacles $O_1, O_2, \ldots, O_m \subseteq W$. Trajectories in an environment can be classified by their topologies into different homotopy classes, which arise from the presence of obstacles in an environment. We start by reviewing some of the standard definitions related to homotopy.

Definition 1 (*Homotopy classes of curves* [17]) Two curves $\gamma_1, \gamma_2 : [0, 1] \to (W - \mathcal{O})$ connecting the same start and end points, are homotopic (or belong to the same *homotopy class*) iff one can be continuously deformed into the other without intersecting any obstacle (see Fig. 2a. See [17, 18] for formal definitions).

For curves in 2-dimensional plane punctured by obstacles, computation of the homotopy class of a given curve can be performed in a relatively simple way [18–22]: We consider *representative points*, ζ_i, inside the ith obstacle O_i [17], and *parallel non-intersecting* rays, $r_1, r_2, \ldots, r_m$, emanating from the obstacles (Fig. 2b). If γ is a given curve whose homotopy class we are trying to identify, we construct

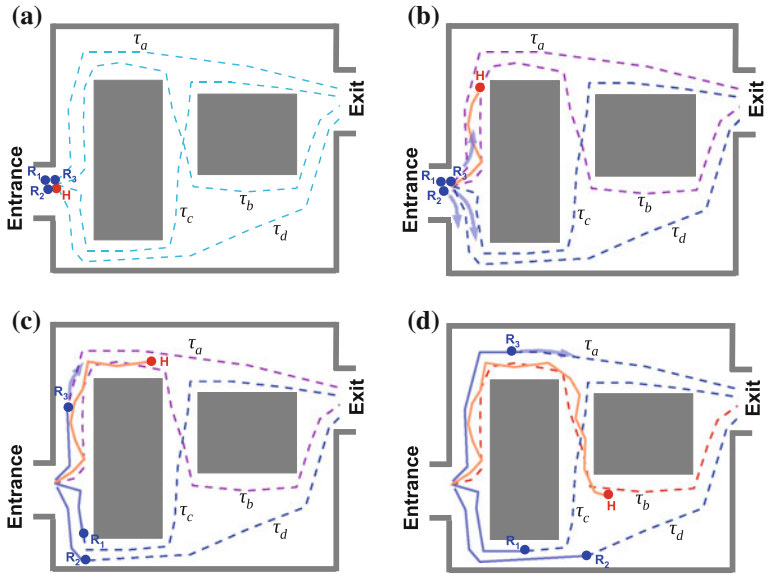


Fig. 2 Homotopy classes and their word representation. **a** γ_1 is homotopic to γ_2 since there is a continuous sequence of trajectories representing deformation of one into the other, but not to γ_3 since it cannot be continuously deformed into any of the other two. **b** ζ_i are representative points inside the obstacles, $O_1, O_2, \ldots, O_m$ (in that order), and r_i, $i = 1, \ldots, m$ are rays emanating from the respective points. The homotopy invariant of this curve γ is $h(\gamma) =$ "$r_2 r_3 r_5 r_6^{-1}$"

a *word* by tracing γ, and consecutively placing the letters of the rays that it crosses, with a superscript of '+1' (assumed implicitly) if the crossing is from left to right, and '−1' if the crossing is from right to left. Thus, for example, the word for γ in Fig. 2b will be "$r_2 r_3 r_4 r_4^{-1} r_5 r_6^{-1}$". We then *reduce* this word by canceling the same letters that appear consecutively but with opposite superscript signs. Thus, the word for γ in Fig. 2b can be reduced to "$r_2 r_3 r_5 r_6^{-1}$". This reduced word representation is a *homotopy invariant* for open curves (with fixed end points), γ, and we will write this as $h(\gamma)$ and call it the "h-signature of γ". The h-signature (reduced word) uniquely identifies the homotopy class of a curve. Note that if the end point of γ coincides with the start point of γ', then $h(\gamma \cup \gamma') = h(\gamma) \circ h(\gamma')$ (where '$\circ$' indicate concatenation of words).

2.2 *h-augmented Graph*

We use a discrete representation of the workspace, W, and construct a graph, G (with vertex set $\mathscr{V}(G)$ and edge set $\mathscr{E}(G)$), by placing a vertex in every accessible discrete cell (cells not intersecting with an obstacle) and by establishing an edge between

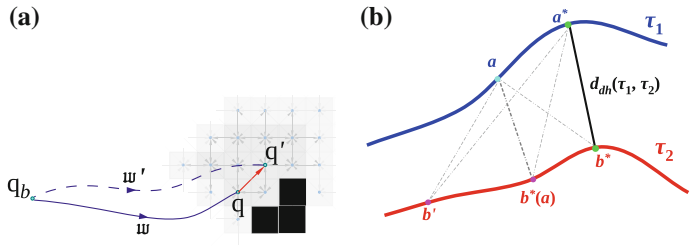


Fig. 3 Preliminaries: h-augmented graph and Hausdorff distance. **a** The h-augmented graph, G_h, is created from the discrete graph representation of the environment, G, so as to incorporate information about the homotopy class of trajectories. **b** The directed Hausdorff distance between τ_1 and τ_2 is determined as follows: Fix a point $a \in \tau_1$, and find its distance from τ_2. The directed Hausdorff distance is then the maximum of this value over all the possible points a on τ_1

the vertices of adjacent cells. While the graph, G, itself can be quite arbitrary, we used a uniform square discretization and an 8-connected graph representation of the environment in all our simulations for simplicity (Fig. 3a).

From such a graph, we construct an h-augmented graph, G_h, for keeping track of the homotopy class of the trajectories. The construction, in brief, is as follows: Vertices in this h-augmented graph, G_h, are of the form $(\mathrm{q}, \mathfrak{w})$ where $\mathrm{q} \in \mathscr{V}(G)$ is a position of an agent in the workspace (as a vertex in the discrete representation graph, $\mathscr{G}$) and $\mathfrak{w}$ is the *word* (i.e. the homotopy invariant) corresponding to the homotopy class of a trajectory leading up to q from a base vertex q_b (all h-signatures of trajectories are computed with respect to a fixed point, called the *base point*, the vertex at which is $\mathrm{q}_b \in \mathscr{V}(G)$). We write the tuple as $\mathrm{v} = (\mathrm{q}, \mathfrak{w}) \in \mathscr{V}(G_h)$, with $\mathrm{v}_b := (\mathrm{q}_b, \text{“ ”})$ being the vertex corresponding to the path of zero length. Thus the h-augmented graph encodes in its vertex set information about homotopy classes of paths, along with agent positions. The connectivity in the h-augmented graph is described as follows: For every directed edge $[\mathrm{q} \rightsquigarrow \mathrm{q}'] \in \mathscr{E}(G)$ (i.e., connecting q to q' in $\mathscr{V}(G)$), and for every vertex of the form $\mathrm{v} = (\mathrm{q}, \mathfrak{w}) \in \mathscr{V}(G_h)$, there exists a vertex $(\mathrm{q}', \ \mathfrak{w} \circ h(\overrightarrow{\mathrm{qq}'}))$ (where $\overrightarrow{\mathrm{qq}'}$ is the line segment corresponding to the edge)—see Fig. 3a. Thus, starting from $(\mathrm{q}_b, \text{“ ”})$, this gives a recipe to construct the h-augmented graph, G_h, incrementally—a construction approach perfectly suited for any graph search algorithm (such as A* and Dijkstra's) involving *expansion* of vertices starting from an initial vertex, so as to find shortest paths leading up to a vertices of the form $(\mathrm{q}_g, *)$—i.e., the goal vertex, q_g, but reached via a specific homotopy class. The cost of an edge in G_h is chosen to be the same as the cost of the projected edge in G. That is, $c_{G_h}([(\mathrm{q}, \mathfrak{w}) \rightsquigarrow (\mathrm{q}', \mathfrak{w} \circ h(\overrightarrow{\mathrm{qq}'})]) = c_G([\mathrm{q} \rightsquigarrow \mathrm{q}'])$ (where c_G and c_{G_h} represent the cost functions in the respective graphs). In our implementation we choose $c_G([\mathrm{q} \rightsquigarrow \mathrm{q}'])$ to be the Euclidean length of the line segment that constitutes the edge, $\overline{\mathrm{qq}'}$. For more details, the reader can refer to similar construction appearing in recent works [23, 24].

2.3 Hausdorff Distance as a Metric on Space of Trajectories

The behavior of a human agent, by nature, can be highly unpredictable. Even if a human is presented with a clear set of trajectories to choose from, he/she may take a trajectory that deviates from the planned ones. In our problem it is critical that the robots quickly understand/estimate which homotopy class the human agents are potentially taking so that the robots can quickly follow the complementary classes. This is achieved by comparing the human's partial trajectory with a set of estimated/baseline candidate trajectories in different homotopy classes connecting the start and the goal locations. The h-signature by itself does not provide adequate information to evaluate distance between candidate trajectories. Rather, this comparison warrants a metric in the space of all trajectories in $(W - \mathcal{O})$. In particular, one can choose the *Hausdorff distance* [25], that is suitable for comparing any two subsets of a metric space.

Definition 2 (*Hausdorff distance* [25]) Consider the free space of the agents, $(W - \mathcal{O})$, equipped with the standard Euclidean metric. The Hausdorff distance, d_H, between two sets $A, B \subset (W - \mathcal{O})$ is then defined as

$$d_H(A, B) = \max(d_{dh}(A, B), d_{dh}(B, A)) \tag{1}$$

where, $d_{dh}(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$ is the directed Hausdorff distance between sets A and B, and $\|a - b\|$ is the Euclidean distance between $a, b \in (W - \mathcal{O})$.

Hausdorff distance, as defined, is a valid metric (satisfying all the *axioms of a metric*) on the space of all subsets of $(W - \mathcal{O})$. In particular, this implies $d_H(A, B) = 0 \iff A = B$. We can thus use this metric to compare two trajectories, τ_1 and τ_2, when viewed as subsets of $(W - \mathcal{O})$. In particular, if τ_H is a trajectory that the human has traversed, and $\{\tau_1, \tau_2, \ldots\}$ are candidate trajectories in different homotopy classes estimated by the robots, then the values of $d_H(\tau_H, \tau_i)$ will help determine which homotopy class the human is following.

Figure 3b illustrates the computation of the directed Hausdorff distance, d_{dh}, for two trajectories τ_1 and τ_2 in an environment. The Hausdorff distance itself is a *symmetrized* version of that distance to satisfy the symmetry property of a metric.

3 Algorithm Design

As described earlier, the objective of this work is to develop a distributed algorithmic framework for autonomous agents in search and rescue operations consisting of a heterogeneous team of humans and robots, taking into account the unpredictability of human agents to efficiently explore an environment via complementary *homotopy*

classes of trajectories. For simplicity, we assume that all trajectories have equal priority and that each robot and each human travel at the same speed, respectively. Priority and speed variation could potentially be considered, if needed, through modification of the cost function to prioritize most promising paths or agents.

As illustrated in Fig. 1, a key critical component in the algorithm design is the ability of the autonomous agents (robots) to identify the intent of the humans. In particular, the robots need to quickly narrow down the set of possible homotopy classes of paths that the humans are potentially taking, and thus follow the complementary classes. Furthermore, they need to monitor whether a human agent is altering his/her behavior (changing the homotopy class that he/she committed to), so that the robots can change their trajectories accordingly. The principal components involved in achieving these are (i) *Human path prediction*, (ii) *Robot path assignment* and (iii) *Human's path history truncation*. The following sub-sections describe these algorithmic components in details.

3.1 Human Path Prediction Algorithm

At the entrance to the environment, the robots compute a set of reference trajectories, $\mathscr{T} = \{\tau_1, \tau_2, \ldots, \tau_p\}$, in p shortest homotopy classes that are to be tentatively pursued by the agents by performing A* search in the h-augmented graph. Ideally, we should choose all the non-looping and non-intersecting homotopy classes in the environment, but for most practical cases it is sufficient to choose $p = \max(P, n + m)$, where n is the number of human agents, m is the number of robot agents, and P is an upper bound on the number of homotopy classes that we compute (determined by our computation capability). We describe the path prediction algorithm for a single human. In the presence of multiple human agents, the algorithm is executed for each of them. Also, in a distributed implementation, the algorithm for predicting each human's path runs independently on each autonomous robot.

At the start, the set of potential paths that the human is following, denoted $\mathscr{S}_0$, is the entire of $\mathscr{T}$. The kth call of the path prediction algorithm computes $\mathscr{S}_k$, the set of potential paths that the human is following at the kth computation step, in a recursive manner. Suppose we computed $\mathscr{S}_k = \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_q}\} \subseteq \mathscr{T}$. The path prediction algorithm at the $(k+1)$th step takes the human's path history (say τ_H) and compares it with the reference trajectories in $\mathscr{T}$ (i.e., computes the Hausdorff distances from each τ_i) to determine the new set $\mathscr{S}_{k+1}$ of homotopy classes that the human is potentially following. The basic algorithm is as follows: Let the distances of the human's path history from the reference trajectories be $d_i := d_H(\tau_H, \tau_i),\ i = 1, 2, \ldots, p$. These distances are normalized by the largest Hausdorff distance out of the most recent potential set of trajectories, $\mathscr{S}_k$, that the human was following to obtain a set of normalized distances: $\overline{d}_i = d_i/D,\ \forall i = 1, 2, \ldots, p$, where $D = \max_{\tau_i \in \mathscr{S}_k} d_H(\tau_H, \tau_i)$. Based on these normalized distances, the objective is to determine if the human's trajectory is *close* to some of the trajectories in $\mathscr{T}$ and *far* from others. This decision ($(k+1)$th path prediction cycle) involves a two-step reasoning:

i. If $\min(\{\overline{d}_1, \overline{d}_2, \ldots, \overline{d}_p\}) \geq \alpha$, where $\alpha \in [0, 1]$ is a parameter encoding the maximum uncertainty tolerated by the user, then the decision cannot be made yet—it is not yet clear what subset of $\mathscr{S}_k$ the human has narrowed down to. Thus $\mathscr{S}_{k+1}$ is not computed, and it is asserted that the $(k+1)$th path prediction cycle is still in progress with $\mathscr{S}_k$ being the set of possible homotopy classes that the human is still following. The robots waiting for the human to make the move keep waiting or continue to follow the same path as before.
ii. If however, $\min(\{\overline{d}_1, \overline{d}_2, \ldots, \overline{d}_p\}) < \alpha$, we update the set of potential paths that the human is following to the set $\mathscr{S}_{k+1} := \{\tau_i \mid \overline{d}_i < \beta \cdot \min(\{\overline{d}_1, \overline{d}_2, \cdots, \overline{d}_p\})\}$, where $\beta \geq 1$ is another parameter. This simply implies that the set of potential homotopy classes that the human is following contains trajectories that are within a distance of at most β times the distance from the closest class. This provides a conservative buffer in the case of very similar paths.

In implementation, following the $(k+1)$th path prediction step, the human broadcasts the h-signatures of the paths in set $\mathscr{S}_{k+1}$ only that are being followed by the human, rather than the full set of vertices describing the predicted path itself. This gives a compact communication protocol purely based on topological information rather than denser metric information. Thus the communication burden is minimized for each human, allowing for more effective and efficient coordination between humans and robots.

Figure 4 shows a simple example of the path prediction algorithm and how the Hausdorff distance is used as a metric to select the set $\mathscr{S}_{k+1}$ of paths for the human's potential trajectory.

3.2 Robot Assignment Algorithm

At the very beginning (start/entrance to the environment), the robots wait for all the humans to make first moves until the humans have a narrowed-down set of possible homotopy classes (i.e., the number of elements in $\mathscr{S}_1$, for each human, has gone below the number of elements in $\mathscr{S}_0 = \mathscr{T}$). Then the robots coordinate among themselves to determine the h-signature of the path (or a set of h-signatures) that each robot should follow. In particular, the cost of a path in a given homotopy class is used to prioritize the assignment of robots to expedite clearing of the environment. This assignment process is run every time a new cycle of path prediction returns a new set of possible homotopy classes that a human is following.

Suppose for the jth human the k_jth cycle of path prediction algorithm returned a new $\mathscr{S}_{k_j}$. The robot assignment algorithm works by first determining the shortest p paths for each robot in the environment, along with the associated path costs. Following which there are two stages in the assignment algorithm:

i. *Choose complementary homotopy classes:* The h-signatures for paths that are not in the set of potential paths that any of the humans are following (i.e. not in $\mathscr{S}_{k_j}$ of any of the humans) are prioritized first—unassigned homotopy classes

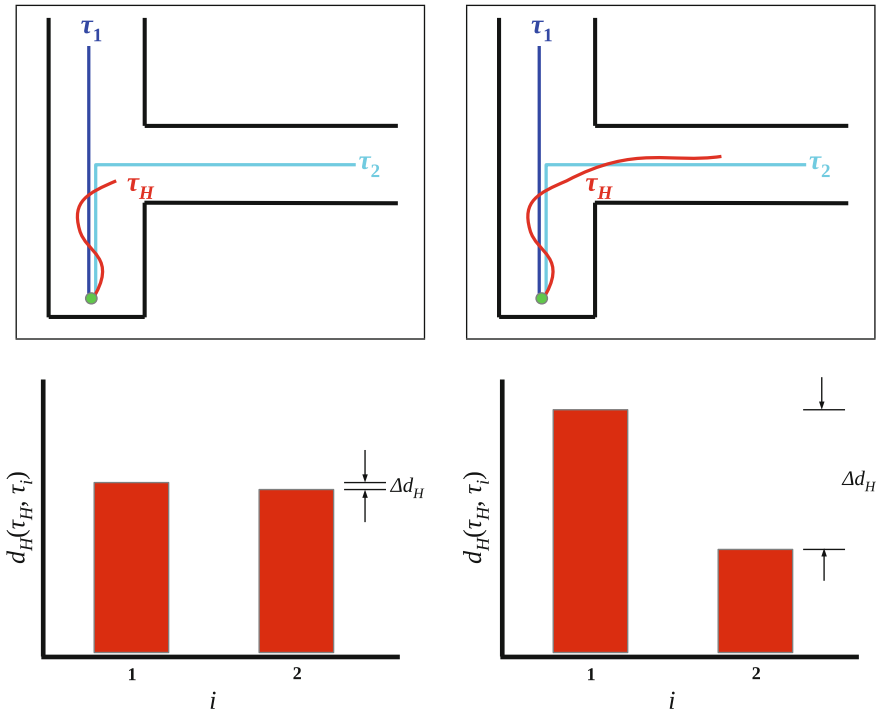


Fig. 4 Initially (*left column*), the human's path τ_H (in *red*) is close to either of τ_1 and τ_2 (i.e. the set of potential paths that the human is following is $\mathscr{S}_k = \{\tau_1, \tau_2\}$). As the human travels further (*right column*), the human's path, τ_H, gets closer to τ_2. In this case, the human path prediction algorithm would update the set of potential homotopy classes of paths to $\mathscr{S}_{k+1} = \{\tau_2\}$. Note how Δd_H (difference between d_1 and d_2) increases indicating a clear demarcation between the distances from the two reference trajectories

get assigned to the robot with the shortest path cost for that homotopy class. This behavior is illustrated by robots R_1 and R_2 in Fig. 1c.

ii. *Tailgate humans with more than one homotopy class in the possible set of homotopy classes:* Once all classes not in any of the human's $\mathscr{S}_{k_j}$ have been assigned to robots, then the remaining robots are assigned to follow human agents with excess elements in their set of possible homotopy classes (the jth human's $\mathscr{S}_{k_j}$). This is the behavior of robot R_3 in Fig. 1c.

This path assignment algorithm is also executed again for groups of tailgating robots every time the human which they are following passes through a junction/branching point (i.e., the path prediction algorithm returns a new set $\mathscr{S}_{k+1}$).

3.3 *Human's Path History Truncation for Robustness to Unpredictable Actions*

This algorithmic component is necessary to incorporate sudden changes in human behavior that contradict the decision made in a prior path prediction step. In case a human turns back and goes past an earlier junction/fork point, and starts following a different homotopy class, the clear demarcation between the Hausdorff distances from the trajectories in $\mathscr{S}$ and those from the other, as was illustrated in Fig. 4, will fade—triggering the 'path history truncation' procedure. In order to figure out which new homotopy class of trajectory the human has taken up, we need to *chop off* the part of the human's path history involving the "U-turn" from the earlier homotopy class, and replace it with a path in the same homotopy class. Consequently, the human path prediction algorithm (say the $(k+1)$th cycle) will be able to identify the new homotopy class the human is following, and compute $\mathscr{S}_{k+1}$ accordingly.

Suppose, for a human, the last path prediction cycle returned $\mathscr{S}_k$. The path truncation algorithm seeks to isolate only the most recent path history for the human so that the path prediction algorithm only uses the most recent, freshest human path data and ignores the convoluted past path behavior. This is achieved by looking at the minimum distance from points on the human's path to reference trajectories in $\mathscr{S}_{k-1}$. Consider a point u' earlier on the human's trajectory (see Fig. 5). The minimum distance of this point from any trajectory of homotopy classes that the human was potentially following before taking the U-turn, $d_{min}(u', \tau_i),\ \forall \tau_i \in \mathscr{S}_k$, can be expected to be low. While the distances from the other homotopy classes, $d_{min}(u', \tau_i),\ \forall \tau_i \in (\mathscr{S}_{k-1} - \mathscr{S}_k)$, can be expected to be high. However, if we consider a point u'' later on the human's trajectory (after the U-turn), this will just be the reverse—$d_{min}(u'', \tau_i),\ \forall \tau_i \in \mathscr{S}_k$ will be high, while $d_{min}(u'', \tau_i),\ \forall \tau_i \in (\mathscr{S}_{k-1} - \mathscr{S}_k)$ will be low. This observation is key in determining the truncation point. In particular, we choose the truncation point to be a point u_{trunc} on the human's trajectory at which the the average of the distances $d_{min}(u_{trunc}, \tau_i),\ \forall \tau_i \in \mathscr{S}_k$ becomes equal to the average of the distances $d_{min}(u_{trunc}, \tau_i),\ \forall \tau_i \in (\mathscr{S}_{k-1} - \mathscr{S}_k)$. After truncation, the human's trajectory is updated by replacing the part before truncation with the shortest path leading to u_{trunc} but in the same homotopy class as the truncated part of the trajectory. This approach will be effective as long as a human is not perpetually indecisive switching between classes forever.

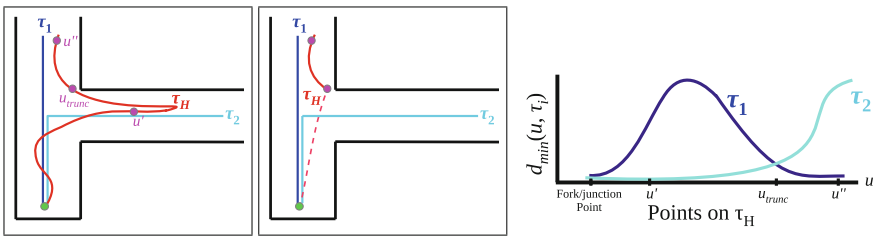


Fig. 5 Identifying the point on the human's trajectory at which to truncate it

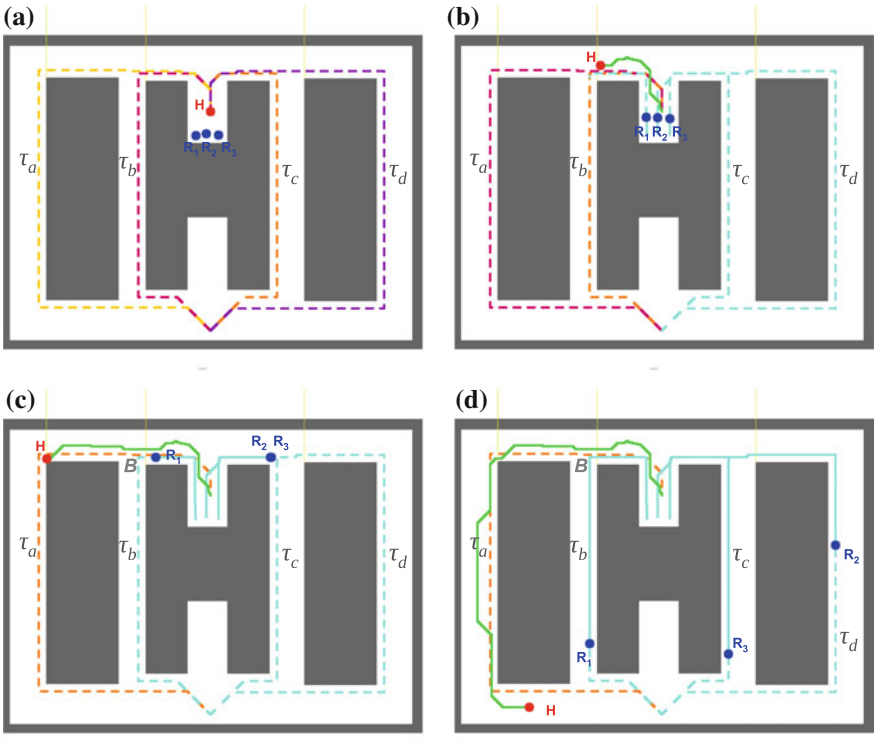


Fig. 6 Collaborative topological exploration in complementary homotopy classes demonstrating how autonomous agents respond to human actions by choosing complementary homotopy classes. **a** Three robots (R_1, R_2, R_3, in *blue*) and one human (H, in *red*) start at the *upper* "cup" of the H-shaped obstacle. They find 4 homotopy classes leading to the goal at the *bottom* of the environment. Optimal trajectories in the different homotopy classes are shown in different colors for easy visualization. At this point the potential set of possible homotopy classes that the human can take is $\mathscr{S}_0 = \{\tau_a, \tau_b, \tau_c, \tau_d\}$. **b** As the human moves forward, its set of possible homotopy classes is narrowed down to $\mathscr{S}_1 = \{\tau_a, \tau_b\}$. The corresponding trajectories are shown in *red* and *orange*. It is determined that homotopy classes of τ_c and τ_d have not been taken up by the human. Thus, homotopy classes of τ_c and τ_d are assigned to robots (R_2 and R_3) with priority. Remaining robot (R_1) is assigned to tailgate human, H, since $\mathscr{S}_1$ contains more than one elements. **c** After crossing a branching point, B, the human commits to homotopy class of τ_a (*orange dotted curve*). So now $\mathscr{S}_2 = \{\tau_a\}$ contains a single element. R_1 thus will choose the complementary class τ_b. Robots R_2, R_3 continues as usual. **d** A final frame showing that the humans and robots followed complementary homotopy classes to reach the goal

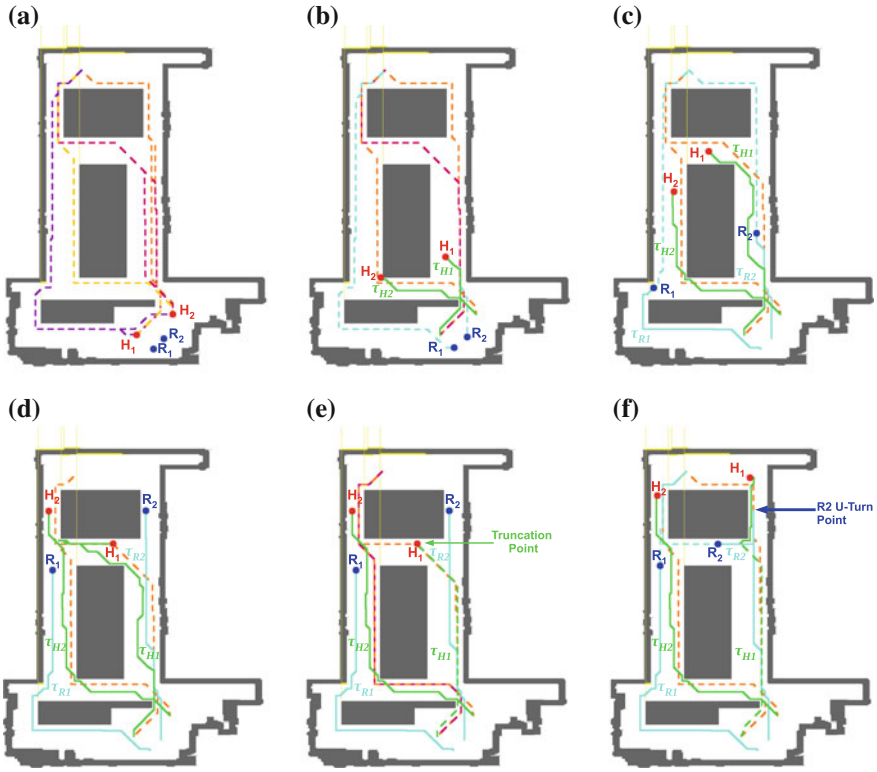


Fig. 7 An example of the human-robot coordinated exploration in an indoor search and rescue scenario, with two robots, two humans and a demonstration of the path truncation algorithm. **a** Two robots (R_1, R_2, in *blue*) and two humans (H_1, H_2, in *red*) start near the *bottom* of the map. They find 4 homotopy classes leading to the goal at the *top* of the map. As in the example with one human, the four shortest paths in different homotopy classes for H_1 and H_2 are displayed. **b** As H_1 and H_2 travel away from the initial junction point, R_1 responds by planning a path in a complementary homotopy class while R_2 tailgates the humans. The planned paths for R_1 and R_2 are shown in *cyan* color. **c** At this point, R_1 and R_2 have already started moving towards the goal. H_1 and H_2 are closer to the goal and appear to only be following one path, respectively, so R_2 goes from tailgating the human to planning a path in the remaining complementary homotopy class. **d** H_1 has turned back from the path it was following previously. H_1 is now traveling towards R_2. The indecisive behavior results in less clarity regarding the human's path behavior. This triggers the *path truncation algorithm* to be executed, so that any future predictions will only focus on the most recent human path data. **e** H_1's path was truncated at the labeled truncated point, eliminating the "U-turn" points from being used as data in the path prediction algorithm. The path before the truncation point is replaced by the shortest path in the same homotopy class as the part of the path that was chopped off (*green dashed curve*). **f** In response to the update in H_1's predicted paths after the path truncation was completed, R_2 makes a "U-turn" to take the path abandoned by H_1—the path to the *left* of the uppermost obstacle. Essentially, H_1 and R_2 have switched places. From this point onwards, all agents travel along these planned paths to the goal point

4 Results

Implementation: The described algorithm was implemented in ROS (Robot Operating System) with human agents simulated through mouse-driven user interface controlled by the authors and autonomous robots navigating using the proposed algorithm. Dynamics or kinematics of the agents were not simulated; however, our implementation is completely distributed, with the agents communicating using h-signatures as compact representations of trajectories. The environment was provided to ROS as a bitmap, with automated identification of connected components of obstacles and placement of representative points. In order to avoid multiplicity of homotopy classes created by small obstacles/noise, a minimum size threshold was placed on the obstacles on which to place *representative points*. Additionally, the obstacles in the bitmap were inflated by the radius of robot to enable collision avoidance and modeling of robots as points in the inflated obstacle map. For the path prediction algorithm we chose the parameters $\alpha = 0.5$ and $\beta = 1.50$ based on experimentation on a benchmark environment. In practice and for simplicity, the path prediction algorithm for each human was implemented on the human agent itself (its processor thread). The predicted paths were communicated to the other agents by reporting the h-signatures of the predicted paths.

Figure 6 shows how three robots and one human split up the process of exploring an environment in four different homotopy classes. Figure 7 demonstrates our algorithm in a more complex indoor environment with two humans and two robots. The example also illustrates the path truncation algorithm. For each of the results, the figure captions describe the algorithm in action. The algorithm was also tested in more complex environments—these results can be viewed at http://www.eecs.berkeley.edu/~govvijay/DARS14.html.

5 Conclusion

A human-robot coordinated exploration problem in context of search and rescue operations is addressed in this paper. The autonomous robots intelligently choose actions to complement the actions of the human agents. In particular, the idea of *complementary homotopy classes* of trajectories help the autonomous agents choose trajectories for fast and efficient exploration. The proposed algorithm consists of prediction of the homotopy classes of the human agents' paths, assignment of complementary paths to the robots, and a truncation algorithm for increased robustness to the indecisive/uncertain behavior of human agents. We demonstrated the practical applicability of the algorithm through ROS simulations with distributed implementation. In the near future, we plan to conduct extensive experiments on real robots and explore the optimal selection of parameters α and β in the path prediction algorithm for an arbitrary environment. Development of additional interfaces for fast and easy communication of intent between the agents is under progress.

Acknowledgments We gratefully acknowledge the support of Army Research Laboratory grant number W911NF-10-2-0016, Air Force Office of Scientific Research grant number FA9550-10-1-0567, and Office of Naval Research grant number N00014-09-1-103. The first author would also like to thank the Rachleff Scholars Program.

References

1. Stachniss, C.: Exploration and mapping with mobile robots. Ph.D. thesis, University of Freiburg, Freiburg, Germany, Apr 2006
2. Bhattacharya, S., Michael, N., Kumar, V.: Distributed coverage and exploration in unknown non-convex environments. In: *Proceedings of 10th International Symposium on Distributed Autonomous Robotics Systems*. Springer, 1–3 Nov 2010
3. Hazon, N., Mieli, F., Kaminka, G.A.: Towards robust on-line multi-robot coverage. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, pp. 1710–1715, May 2006
4. Rekleitis, I., New, A.P., Rankin, E.S., Choset, H.: Efficient boustrophedon multi-robot coverage: an algorithmic approach. Ann. Math. Artif. Intell. **52**(2–4), 109–142 (2008)
5. Zheng, X., Koenig, S., Kempe, D., Jain, S.: Multirobot forest coverage for weighted and unweighted terrain. IEEE Trans. Robot. **26**(6), 1018–1031 (2010)
6. Bhattacharya, S., Ghrist, R., Kumar, V.: Multi-robot coverage and exploration on Riemannian manifolds with boundary. Int. J. Robot. Res. **33**(1), 113–137 (2014). doi:10.1177/0278364913507324
7. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**, 129–137 (1982)
8. Cortez, J., Martinez, S., Bullo, F.: Spatially-distributed coverage optimization and control with limited-range interactions. ESIAM: Control Optim. Calc. Var. **11**, 691–719 (2005)
9. Cortez, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. and Automat. **20**(2), 243–255 (2004)
10. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms. Applied Mathematics Series. Princeton University Press, Princeton (2009)
11. Sariel-Talay, S., Balch, T.R., Erdogan, N.: Multiple traveling robot problem: a solution based on dynamic task selection and robust execution. IEEE/ASME Trans. Mechatron. **14**(2), 198–206 (2009). April
12. Carlin, A., Ayers, J., Rousseau, J., Schurr, N.: Agent-based coordination of human-multirobot teams in complex environments. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Industry Track, AAMAS'10, pp. 1747–1754, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2010)
13. Kehagias, A., Hollinger, G., Singh, S.: A graph search algorithm for indoor pursuit/evasion. Math. Comput. Modell. **50**(910), 1305–1317 (2009)
14. Choset, H., Burdick, J.: Sensor-based exploration: the hierarchical generalized Voronoi graph. Int. J. Robot. Res. **19**(2), 96–125 (2000)
15. Tully, S., Kantor, G., Choset, H.: A unified Bayesian framework for global localization and SLAM in hybrid metric/topological maps. Int. J. Robot. Res. (2012)
16. Kim, S., Bhattacharya, S., Ghrist, R., Kumar, V.: Topological exploration of unknown and partially known environments. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3851–3858, Nov 2013
17. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Auton. Robots 1–18, (2012). doi:10.1007/s10514-012-9304-1
18. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2001)

19. Grigoriev, D., Slissenko, A.: Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In: ISSAC'98: Proceedings of the 1998 international symposium on symbolic and algebraic computation, pp. 17–24, New York, NY, USA. ACM (1998)
20. Hershberger, J., Snoeyink, J.: Computing minimum length paths of a given homotopy class. Comput. Geom. Theory Appl **4**, 331–342 (1991)
21. Tovar, B., Cohen, F., LaValle, S.M.: Sensor beams, obstacles, and possible paths. In: Workshop on the Algorithmic Foundations of Robotics (2008)
22. Narayanan, V., Vernaza, P., Likhachev, M., LaValle, S.M.: Planning under topological constraints using beam-graphs. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 431–437. IEEE (2013)
23. Kim, S., Bhattacharya, S., Heidarsson, H., Sukhatme, G., Kumar, V.: A topological approach to using cables to separate and manipulate sets of objects. In: Proceedings of the Robotics: Science and System (RSS), Syndey, Australia, 24–28 June 2013
24. Kim, S., Bhattacharya, S., Kumar, V.: Path planning for a tethered mobile robot. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2014) (Accepted. To appear)
25. Gromov, M., Lafontaine, J., Pansu, P.: Metric Structures for Riemannian and Non-Riemannian Spaces. Progress in Mathematics. Birkhäuser, Basel (1999)

A Repartitioning Algorithm to Guarantee Complete, Non-overlapping Planar Coverage with Multiple Robots

Kurt Hungerford, Prithviraj Dasgupta and K.R. Guruprasad

Abstract We consider the problem of coverage path planning in an initially unknown or partially known planar environment using multiple robots. Previously, Voronoi partitioning has been proposed as a suitable technique for coverage path planning where the free space in the environment is partitioned into non-overlapping regions called Voronoi cells based on the initial positions of the robots, and one robot is allocated to perform coverage in each region. However, a crucial problem arises if such a partitioning scheme is used in an environment where the location of obstacles is not known a priori—while performing coverage, a robot might perceive an obstacle that occludes its access to portions of its Voronoi cell and this obstacle might prevent the robot from completely covering its allocated region. This would either result in portions of the environment remaining uncovered or requires additional path planning by robots to cover the disconnected regions. To address this problem, we propose a novel algorithm that allows robots to coordinate the coverage of inaccessible portions of their Voronoi cell with robots in neighboring Voronoi cells so that they can repartition the initial Voronoi cells and cover a set of contiguous, connected regions. We have proved analytically that our proposed algorithm guarantees complete, non-overlapping coverage. We have also quantified the performance of our algorithm on e-puck robots within the Webots simulator in different environments with different obstacle geometries and shown that it successfully performs complete, non-overlapping coverage.

Keywords Multi-robot systems · Coverage path planning · Voronoi partitioning

K. Hungerford · P. Dasgupta (✉)
Computer Science Department, University of Nebraska, Omaha, USA
e-mail: pdasgupta@unomaha.edu

K. Hungerford
e-mail: khungerford@unomaha.edu

K.R. Guruprasad
Department of Mechanical Engineering, National Institute of Technology, Mangaluru, Karnataka, India
e-mail: krgprao@gmail.com

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_3

1 Introduction

Coverage path planning is a central aspect of multi-robot systems where the objective is to completely cover the surface area of an environment using multiple robots. Robotic coverage is used in several application domains of robots such as unmanned search and rescue, clearing an area of landmines, inspecting the health of engineering structures, as well as in civilian applications such as automated lawn mowing and vacuum cleaning. Using multiple robots for area coverage instead of a single robot offers several advantages such as reducing the time required to complete the environment's coverage and improving the robustness of the system to failure of a single or a few robots. However, using multiple robots also introduces the overhead of coordination between robots to avoid collisions and perform non-overlapping coverage. An attractive technique to implement non-overlapping coverage between robots is to partition the free space of the environment into disjoint regions or cells that can then be covered by robots [1–4]. In most of these partitioning-based coverage techniques, the cellular partitions are not changed once they have been determined. However, if the obstacles inside the environment are initially unknown to the robots, a robot might discovers that a cell is occluded by an obstacle while performing coverage. As shown in Fig. 1a, a robot then has to use path planning techniques to explore paths to reach the cell's occluded part. In a multi-robot coverage scenario, the path planning technique to reach the occluded portions of a cell involves significant computation and coordination between the robots [3], which might result in increased battery expenditure and completion time for the coverage. Therefore, it makes sense to investigate techniques that can reduce or avoid these path planning costs for robots by adaptively repartitioning cells and reallocating the repartitioned portions, so that other robots can cover the repartitioned cell with little overhead for navigation planning.

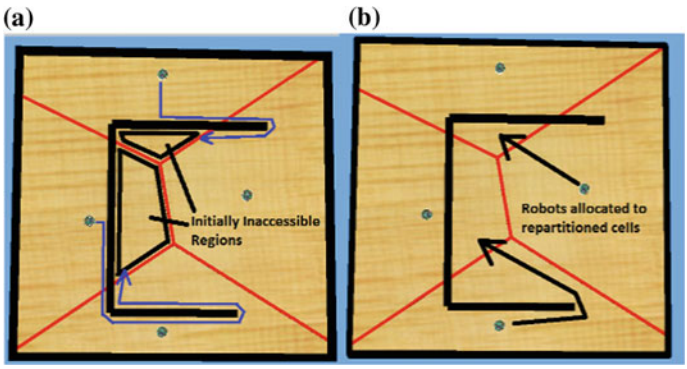


Fig. 1 **a** The Voronoi cells of two robots are partially inaccessible due to obstacles. The *blue solid arrows* show the path taken by a robot to reach the inaccessible portions of its cell using a bug-like path planning algorithm. **b** Robots coordinate with each other to repartition the initial Voronoi cells so that each robot has a contiguous region to cover

Our research in this paper is based on the key insight that when the initial partition of the environment is done equitably between robots, exactly one robot occupies a cell. Then, even if the cell that a robot is covering gets disconnected due to obstacles, because the free space is connected, the inaccessible portion of the cell must be adjacent to at least one of the neighboring cells and accessible to the robot in that cell. Consequently, the robot performing coverage in the adjacent neighboring cell could be requested to augment its coverage with the inaccessible portion of the disconnected cell, as shown in Fig. 1b. Based on this insight, we first partition the environment into complete non-overlapping cells using Voronoi partitioning [5] and then propose a novel algorithm called *Repart-Coverage*, where each robot performs boundary coverage of its initially allocated cell or region and then uses a low-overhead coordination protocol with other robots to systematically repartition only those portions of its cell that are inaccessible due to obstacles. We have shown analytically that our proposed technique guarantees complete, non-overlapping coverage. We have also verified the performance of the *Repart-Coverage* algorithm on simulated e-puck robots within the Webots simulator for different environments and different obstacle geometries and quantified its performance in terms of the areas of coverage regions and distances traveled by the different robots due to repartitioning of their initial cells.

2 Related Work

Coverage path planning has been a central topic in robot motion and an excellent survey is given in [6], including both single and multi-robot coverage. For multiple networked robots performing distributed coverage, the coordination strategies that have been proposed can be divided into two broad categories. In the first category, robots share maps of covered regions with each other while they perform coverage so that they can coordinate their movements to avoid each others' regions. In most of these techniques, the environment is divided into grid-based cells corresponding to the footprint of a robot. Robots then use different techniques to avoid repeated coverage such as sensing and avoiding already-covered neighboring cells [7], recording the regions covered by each robot as a coverage tree [8] and communicating the boundaries of covered regions between the robots, and, using a negotiation protocol along with a distance-based objective function to select regions to cover for different robots [9]. In [3], the authors proposed a technique called multi-robot Boustrophedon decomposition where the robots decompose the environment into cells in an online manner while performing coverage. Robots use two different roles - boundary coverage and area coverage. A pair of boundary coverage robots move in tandem along two parallel but opposite boundaries of the environment and infer about the presence of obstacles when the line of sight between them gets blocked. This information is used to define cell boundaries for subsequent coverage by the area coverage robots. The algorithm can guarantee complete, non-overlapping coverage, but the robots have to

use complex calculations and tight coordination to guarantee that cell boundaries are correctly identified and multiple robots are not assigned to cover the same cell.

In contrast, in the second category of coverage coordination, the environment is partitioned into non-overlapping cells based on the initial positions of robots using strategies such as polygonal decomposition [2], Voronoi partitioning [4, 5, 10], etc. Recently, while extending this approach, Breitenmoser et al. have proposed an algorithm where robots initially partition the environment using Voronoi cells and start navigating towards target locations while continuously adapting the partitions and refining the target locations as they discover obstacles [11]. In [4, 12], the authors have proposed a multi-robot coverage technique where each robot communicates its position while it moves and dynamically adapts the partitions with neighboring robots to guarantee complete, non-overlapping partition of the environment. In contrast to our work, they do not explicitly address situations that prevent the complete coverage of a Voronoi cell assigned to a robot when a portion of the cell becomes inaccessible to the robot due to obstacles. Since the focus of our paper is on partitioning the environment for coverage, we use a boundary coverage algorithm called Egress [13] that enables a robot to determine and follow the boundary of its currently assigned region; we assume that suitable techniques for covering the internal area of a region such as ladder search [2] or spanning tree coverage(STC) [14] are utilized by the robot after it has determined the boundary of the region it has to cover. Also, in the rest of the paper we have used the term coverage to refer to boundary coverage.

3 Problem Formulation

Let $Q \subset \mathbb{R}^2$, a convex polygon, represent a region occupied by a set of obstacles O. Let $Q_{\text{free}} = Q \backslash O$, denote the free space within Q. We assume Q_{free} to be a topologically connected set. Our objective is to perform complete, non-overlapping coverage of the region $Q \backslash O$, using N autonomous mobile robots, each equipped with a coverage tool. Let $\mathcal{P}(t) = \{p_i(t) \in Q, i \in I_N\}$, where $p_i(t)$ denotes the position of the ith robot at time t.[1]

The *Voronoi partition*, generated by $\mathcal{P}$ is the collection $\{V_i(\mathcal{P})\}_{i \in I_N}$ with,

$$V_i(\mathcal{P}) = \left\{ q \in Q | \parallel q - p_i \parallel \leq \parallel q - p_j \parallel, \forall j \in I_N \right\} \tag{1}$$

The Voronoi partition induces an undirected graph known as Delaunay graph, $\mathcal{G}_D$, where two nodes $i, j \in I_N$ are neighbors if the intersection of corresponding Voronoi cells V_i and V_j is a line segment. The set of neighbors of the node i is denoted as $N(i)$; for brevity we assume $N_i = \mid N(i) \mid$. Let B_{ij} denote the perpendicular bisector

[1] Robots can assume a well-distributed initial configuration in case their initial positions are close to each other using techniques in [11, 15].

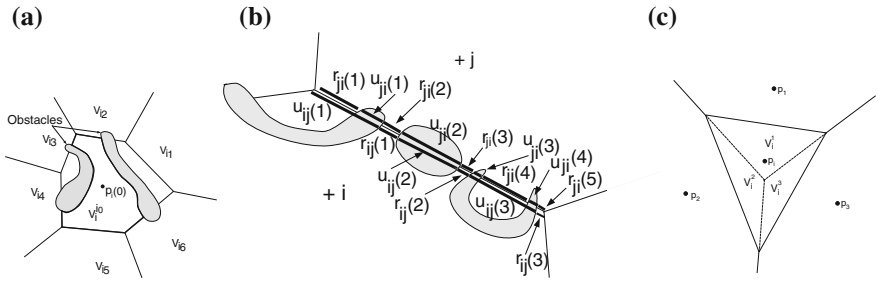


Fig. 2 **a** The region bounded by *dark lines* is $V_i^{i_0}$. **b** Illustration of A_{ij}^b and A_{ij}^f. **c** Illustration of V_i^j when V_i is repartitioned between robots $j \in N(i)$

of line joining $p_i(0)$ and $p_j(0)$ and let $A_{ij} \subseteq B_{ij}$ represent the common boundary between V_i and V_j. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ be a partition of Q_{free}. Let $S_i \in 2^{\mathcal{C}}$, $i \in I_N$, and each S_i, $i \in I_N$ is made up of contiguous cells from $\mathcal{C}$, that is, $\bigcup_{C_j \in S_i} C_j$ is a (topologically) connected set.

Distributed spatial partitioning problem: For each $i \in I_N$, the ith robot should construct S_i, a contiguous collection of topologically connected cells, such that the collection $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ partitions Q_{free}.

3.1 Definitions and Notations

Let $V_i^{i_0} \subset V_i$ be the subset of V_i containing $p_i(0)$. If there are no obstacles within V_i, then $V_i^{i_0} = V_i$. The boundary of $V_i^{i_0}$ is made up of portions of A_{ij} and obstacle boundaries. A point $q \in A_{ij}$ is reachable to robot i from $p_i(0)$, if $q \in V_i^{i_0}$, and unreachable otherwise. Figure 2a illustrates $V_i^{i_0}$ with an example.

Let $A_{ij}^b = \{u_{ij}(k) | A_{ij} \supset u_{ij}(k) \notin V_i^{i_0}\}$, where $u_{ij}(k)$s are mutually disjoint convex sets, representing parts (line segments) of A_{ij} that are not reachable (blocked by obstacles) to the robot i. Similarly, let $A_{ij}^f = \{r_{ij}(k) | A_{ij} \supset r_{ij}(k) \in V_i^{i_0}\}$, where $r_{ij}(k)$s are mutually disjoint convex sets, representing parts (line segments) of A_{ij} that are reachable (not blocked by obstacles) to the robot i. See Fig. 2a for illustration. Note that $A_{ij}^f = A_{ij} \backslash A_{ij}^b$.

Let $N^{fb}(i) = \{j | A_{ij}^b = A_{ij}\} \subset N(i)$. When $j \in N^{fb}(i)$, entire A_{ij} is unreachable to the robot i; then the robot i can not enter V_j without entering V_k, for some $k \notin \{i, j\}$. Let $N^b(i) = \{j | A_{ij}^b \neq \emptyset\} \subseteq N(i)$. Note that $N^{fb}(i) \subset N^b(i) \subseteq N(i)$.

Note that $A_{ij} = A_{ij}^b \cup A_{ij}^f$, and $A_{ij}^b \cap A_{ij}^f = \emptyset$, thus A_{ij}^b and A_{ij}^f partition A_{ij}. If $A_{ij} = A_{ij}^b$ (that is, $A_{ij}^f = \emptyset$), then we say that A_{ij} is impermeable to the robot i. If $A_{ij} = A_{ij}^f$, then we say that A_{ij} is fully permeable to the robot i. If $A_{ij}^b \neq \emptyset$ and

$A^f_{ij} \neq \emptyset$, then A_{ij} is partially permeable to the robot i. Note that $A_{ij} = A_{ji}$, However $A^b_{ij} \neq A^b_{ji}$, and $A^f_{ij} \neq A^f_{ji}$, in general.

Let $V^i_j \subset V_j$, for $j \in N(i)$, be a portion of V_j that would have been part of V_i with node set $I_N \backslash \{j\}$. See Fig. 2c for illustration. $V^i_j = V_j \cap ({}^j\tilde{V}_i)$, where ${}^j\tilde{V}_i$ is the Voronoi cell of i with nodes $I_N \backslash \{j\}$, or just $N(j)$. Each portion of $V_i \backslash V^{i_0}_i$, is part of V^j_i, for some $j \in N(i)$. If A_{ij} is fully impermeable to the robot i, that is, $A_{ij} = A^b_{ij}$, then i will not be able to reach V^i_j.

4 Distributed Spatial Partitioning

In this section, we explain the proposed distributed spatial repartitioning scheme. The ith robot first explores $V^{i_0}_i$ and obtains the following information: (i) V_i, $N(i)$, $p_i(0)$, $p_i(t)$, $p_j(0) \in Q, \forall j \in I_N$ the position of itself and initial positions of all other robots; (ii) A_{ij}, A^b_{ij}, and A^f_{ij}, for each $j \in N(i)$; (iii) the sets $N^{fb}(i) \subset N^b(i) \subset N(i)$, and iv) $V^{i_0}_i$. Now,the robot broadcasts the following information: A^b_{ij}, $A^f_{ij}, \forall j \in N(i)$, and the sets $N^{fb}(i)$, $N^b(i)$, and $N(i)$. This communication is required only at the beginning of the distributed spatial partitioning.

Now the robot uses the available information and further exploration when required, to decide on the additional regions that need to be covered by it. The free regions in $V_j \backslash V^j_j$, $j \in I_N$ can not be covered by robot j and hence need to be covered by other robots. These regions are divided into *patches*. A patch is defined as a connected subset of Voronoi cells. Each patch is bounded by obstacles and/or line segments of B_{jk}, for some $j, k \in I_N$. The ith robot maintains a set S_i of patches it should cover. It is clear that $V^{i_0}_i$ is a patch in S_i. The ith robot adds to S_i patches in $(V_j \backslash V^j_j)_{\text{free}}$—the portion of obstacle free region with V_j, not accessible directly to the robot j, $j \in I_N \backslash \{i\}$. We say that two patches U and W are adjacent, if $U \cap W$ contains a line segment in B_{jk} (not necessarily A_{jk}), for some $j, k \in I_N$ (j and k are not necessarily neighbors)[2]. The significance of two patches U and W being adjacent is that a robot can move freely between these patches. The patches are created as robots explore the regions to be covered. We will discuss the process of constructing S_i in steps.

Scenario i. Patches in V^i_j, $j \in N(i)$ The robot i enters $V^i_j \subset (V_j \backslash V^{j_0}_j)_{\text{free}}$, $j \in N(i)$, if and only if $\exists l \in \{1, 2, \ldots, | A^f_{ij} |\}$, s.t. $r_{ij}(l) \cap A^b_{ji} \neq \emptyset$. This condition is illustrated in Fig. 3a. This patch, say U_1, is adjacent to $V^{i_0}_i$ and is added to S_i.

Scenario ii. Patches in V^k_j, k, $j \in N(i)$, $k \in N(j)$: If the robot i enters a patch $U_1 \subseteq V^i_j$, it explores U_1. If a portion U_2 of V^k_j, $k \in N(i) \cap N(j)$ is adjacent to U_1, then robot i will find out if k can reach this portion of V^k_j. Otherwise, this portion of

[2] As U and W belong to free space, $U \cap W$ is either $\emptyset$ or a permeable line segment.

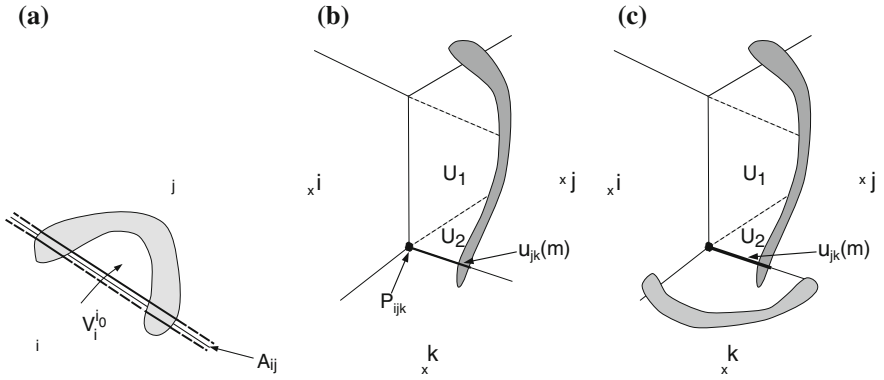


Fig. 3 **a** Robot i can help robot j to cover ${}^{i}(V_j^i)_{\text{free}}$. *Thick solid* and *dashed lines* represent the blocked and free components of A_{ij} respectively. **b–c** Conditions i checks to find out if it has to help a common neighbor k to cover a portion of the region $(V_j^k)^f$

V_j^k will be added S_i. We will discuss the situations in which robot i should or should not cover a patch in V_j^k. Let P_{ijk} be the vertex common to V_i, V_j, and V_k.

1. Consider a scenario, as illustrated in Fig. 3c, where $U_1 \cap V_j^k$ is a single line segment and $P_{ijk} \in U_1$. Let $u_{jk}(m) \in A_{jk}^b$ contains P_{ijk} (Such $u_{jk}(m)$ exists as P_{ijk} is assumed to be part of U_2 adjacent to U_1).
   1a. If $u_{jk}(m) \cap A_{kj}^f \neq \emptyset$, as illustrated in Fig. 3c, a, then k can reach U_2, and hence i will not cover it.
   1b. Otherwise, as illustrated in Fig. 3c, b, k can not reach U_2 and i should cover it. The robot i can check if $P_{ijk} \in U_1 \cap U_2$, and if $U_1 \cap U_2$ is a single connected piece, while physically exploring the boundary of U_1.
2. Consider a scenario, $U_1 \cap V_j^k$ is a not a single line segment or $P_{ijk} \notin U_1$, as illustrated in Fig. 4. In such a scenario, robot i will not be able to decide if U_2 needs to be added to S_i or not only based on available information. The patch U_2 is added to S_i, only if, while physically exploring the boundary of U_2, the robot i reaches a portion of A_{kj}^f.

Remark 1 Note that the robot i physically explores the boundary of a patch W which is adjacent to $U \in S_i$, only when the information about free and blocked regions of Voronoi cell boundaries (A_{ij}) is not sufficient to make a decision as to if W needs to be added to S_i. Such an exploration is local to the robot i and it does not affect the decisions of other robots. This can be observed from the illustrations in Fig. 4. The patch U_2 is added to S_i (Fig. 4a and b) when i concludes that $U_2 \notin S_k$, and is not added to S_i (Fig. 4c) when $U_2 \in S_k$. This ensures that the patch U_2 is covered exactly by one robot.

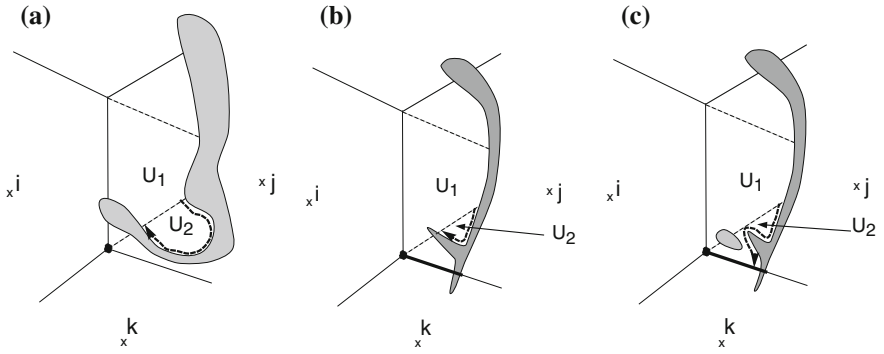


Fig. 4 The robot i explores $U_2 \subseteq V_j^k$ to check if k can reach it. The exploration path is shown with *dark dashed line* ending with an arrow. **a** U_2 is unreachable to the robot k, hence it is added to S_i. **b** Though U_2 is reachable to the robot k, it has to pass through U_1 to reach it. In other words, U_2 is not adjacent to V^k ($V^k \cap U_2 = \emptyset$). Thus $U_2 \in S_i$. **c** The robot i reaches a point on A_{kj}^f while exploring U_2 and hence $U_2 \notin S_i$ (as $U_2 \in S_k$)

Further, it can be noted the scenarios discussed above are exhaustive.

Scenario iii. Patches in V_j^l, $j \in N(i), l \in N(j), l \notin N(i)$ If $V_j^i \supset U_1 \in S_i$ and $U_2 \subset V_j^l$, s.t. U_2 is adjacent to U_1, then robot i has to make a decision on adding U_2 to S_i.

(1) If $V_j^l \backslash V_j^{j_0}$ is not accessible to robot l (based on A_{lj}^f and A_{jl}^b, discussed in scenario(i)), then U_2 is added to S_i. Such scenarios are illustrated in Fig. 5a and b.
(2) Otherwise, the robot i should explore[3] the boundary of entire portion of $V_j^l \backslash V_j^{j_0}$ connected to U_2. Only if no portion of A_{lj}^f is reached during this exploration, U_2 is added to S_i. Figure 5c shows a scenario where U_2 is not added to S_i after a point on A_{lj}^f is reached while exploration (indicating that $U_2 \in S_l$). Figure 5d shows a scenario where U_2 is added to S_i after the robot i fails to reach any point on A_{lj}^f while exploring (indicating that $U_2 \notin S_l$).

Scenario iv. A patch in $V_i \backslash V_i^{i_0}$ While the robot i is in $U_1 \subset V_i \backslash V_j^i$ for some $j \in N(i)$ and $U_1 \in S_i$, and a patch $U_2 \subset V_i \backslash V_i^{i_0}$ is adjacent to U_1, U_2 is added to S_i only if there is no $k \in N(i)$ such that U_2 is adjacent to $V_k^{k_0}$.A condition under which the robot i adds U_2 to S_i is illustrated in Fig. 5e. If U_2 is adjacent only to U_1, that is $P_{ijk} \notin U_1 \cap U_2, k \in N(i) \cap N(j)$, then U_2 is added to S_i.

[3]Remark 1 is also applicable here.

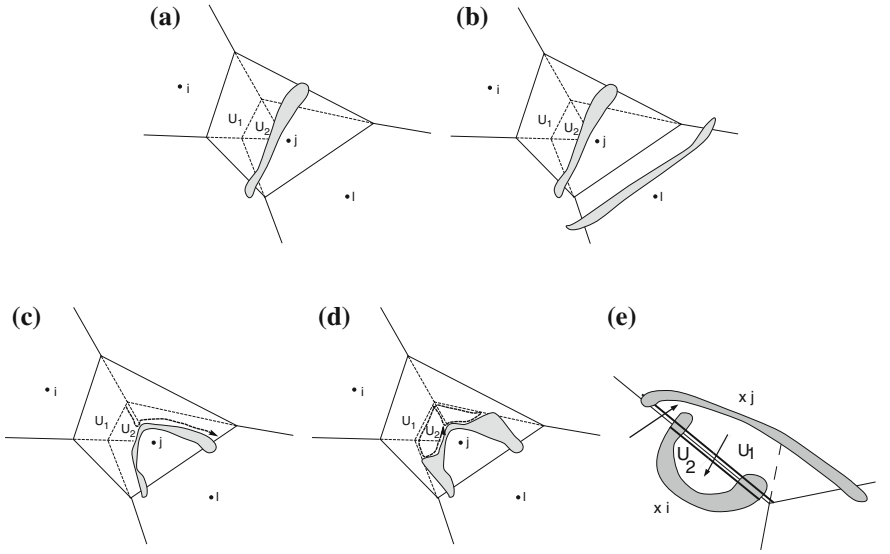


Fig. 5 In situations such as illustrated in **a** and **b** robot l will not enter any portion of V_j^l and hence robot i adds the region $U_2 \subset V_j^l$ to S_i. The robot explores a portion of V_j^l adjacent to U_1 and **c** reaches a point on A_{lj}^f, indicating that the robot l can reach and cover U_2, thus $U_2 \notin S_i$, **d** does not reach any point on A_{lj}^f, indicating that the robot l cannot reach U_2 and hence it is added to S_i. The exploration path is shown in *dark dashed line*. **e** Situations in which the robot i covers a part of $V_i \backslash V_i^{i0}$

Scenario v. Beyond neighbors and neighbors of neighbors: Robot i continues looking for new patches and adds them to S_i based the following principle: Let $U \in S_i$, and W is adjacent to U. Now, W is added to S_i if either, W can not be reached by any other robot, or, if robot i is closer to W than any other robot.

The process continues until the robot finds no more adjacent patches to be added. At this point robot i performs area coverage of the current patch and returns to the previous patch. It finds if there are any new adjacent patches to be added; if not, it performs area coverage of this patch and goes back to the previous patch. Once robot i reaches the patch from which it was initially placed, $V_i^{i_0}$, it performs area coverage of that patch and stops.

4.1 Analytical Results

Lemma 1 *Let $V_{-i} \subset Q_{free} \in V_i \backslash V_i^{i_0}$ denote a region inaccessible to robot i. Then V_{-i} must be topologically connected to robot $j \in N(i)$'s Voronoi cell, V_j.*

Proof (*by contradiction.*) Assume that V_{-i} is not topologically connected to V_j. Suppose the only Voronoi boundary V_{-i} intersects is A_{ij}. There can be two cases of robot i's blocked boundary A^b_{ij} that resulted in V_{-i}:

Case 1. $A^b_{ij} = A^b_{ji}$. Since, $A_{ij} = A_{ji}$, this case implies $A^f_{ij} = A^f_{ji}$. Also, since the only Voronoi boundary V_{-i} intersect is A_{ij}, $V_{-i} \cap A_{ij} = A^b_{ij}$. Substituting this value of expression in the definition of free and blocked boundaries of A_{ji} and noting that $A^b_{ij} = A^b_{ji}$ and $A_{ij} = A_{ji}$, we get: $A^b_{ji} \cap A^f_{ji} = \{\emptyset\}$, or, $V_{-i} \cap A_{ij} \cap A^f_{ji} = \{\emptyset\}$, or, $V_{-i} \cap A_{ji} \cap A^f_{ji} = \{\emptyset\}$. But $A_{ji} \cap A^f_{ji} = A^f_{ji}$ (from the definition of A^f_{ji}). Therefore, we get, or, $V_{-i} \cap A^f_{ji} = \{\emptyset\}$. From the definition of a patch given in Sect. 4, a patch is bounded either by obstacles or by A_{ij}. Since V_{-i} is not accessible from V_i, it is bounded by obstacles from the side of V_i. And, $V_{-i} \cap A^f_{ji} = \{\emptyset\}$ implies it is not accessible (bounded by obstacles) from the side of V_j also. Since A_{ij} is the only Voronoi boundary intersecting V_{-i}, V_{-i} is bounded from all sides by obstacles. In other words, $V_{-i} \not\subset Q_{free}$, which contradicts our assumption.
Case 2. $A^b_{ij} \neq A^b_{ji}$. Suppose $A^b_{ji} \subset A^b_{ij}$.[4] Then the portion of A^b_{ij} that is not shared with A^b_{ji}, must be free (accessible) on the side of V_j; otherwise it would have been part of A^b_{ji}. The portion of boundary that is free only on side of V_j but not of side of V_i is $A^f_{ji} \backslash A^f_{ij}$. That is, $A^f_{ji} \backslash A^f_{ij} \subseteq A^b_{ij} \backslash A^b_{ji}$, or, $A^f_{ji} \backslash A^f_{ij} \subseteq A^b_{ij}$ (since $A^b_{ji} \subset A^b_{ij}$). This implies that the patch V_{-i} is topologically connected to V_j through A^f_{ji}, which contradicts our assumption that V_{-i} is not topologically connected to V_j. Hence proved.

Lemma 2 *A region $V_{-i} \subset Q_{free} \in V_i \backslash V_i^{i_0}$ that is inaccessible to robot i, must be topologically connected to V_j, $j \in I_N$.*

Proof The proof of Lemma 1 can be easily extended to a scenario where V_{-i} intersects more than one neighbor in $N(i)$ by considering the blocked boundary with each neighbor disjointly. For a more general case where V_{-i} is topologically connected only to $N^{(k)}(i)$, the kth hop Voronoi neighbor of i, $k > 1$ (scenarios iv. and v. in Sect. 4), the proof of Lemma 1 still holds between robots i and $j \in N^k(i)$. Varying k over 1 through the maximum hops between the farthest Voronoi cell from i, we get $N^k(i) = I_N$; hence proved.

Theorem 1 *The proposed distributed partitioning and coverage scheme ensures complete coverage of the free space.*

Proof By Lemmas 1 and 2, there must be a robot $j \in I_N$ whose Voronoi cell V_j is topologically connected to V_{-i}. This ensures that for every robot $i \in I_N$, the free space in its Voronoi cell V_i denoted by $Q_{free} \cap V_i$ gets covered by itself or by one or more robots in $j \in I_N$. The total region covered by all robots in I_N is then given

[4] A similar result can be proved for $A^b_{ij} \subset A^b_{ji}$ by interchanging indices i and j.

by $\cup_i (Q_{free} \cap V_i) = Q_{free} \cap (\cup_i V_i) = Q_{free} \cap Q$ (from definition of Voronoi cell) $= Q_{free}$ (since $Q_{free} \subseteq Q$). Hence proved.

Theorem 2 *The proposed distributed partitioning and coverage scheme achieves non-overlapping coverage.*

Proof The proof follows from the construction of patches using Voronoi cell boundaries. From definition, a patch between V_i and V_j is bounded either by obstacles or by the bisector line B_{ij} between robots i and j's initial positions $p_i(0)$ and $p_j(0)$. The Voronoi partitioning is done only once at the beginning, and by definition (Eq. 1) guarantees non-overlapping Voronoi cells. Since there is only one robot per Voronoi cell, the coverage of the initial Voronoi cell ($V_i^{i_0}$) is done only by robot i. When a region $V_{-i} \in V_i \backslash V_i^{i_0}$ is inaccessible from $V_i^{i_0}$, if V_{-i} is adjacent to only one other Voronoi cell V_j then only robot j covers V_{-i}. On the other hand, if V_{-i} is adjacent to more than one Voronoi cell $V_{j1}, V_{j2}, \ldots$ then each pair of robots j_a and j_b divide the region of V_{-i} into patches S_{j_a}, S_{j_b} by extending their bisector lines B_{j_a,j_b}. This construction ensures that $S_{j_a} \cap S_{j_b} = \{\emptyset\}$, or, patches S_{j_a}, S_{j_b} are non-overlapping; patch S_{j_k} is covered only by robot j_k. Therefore, for every robot i, $V_i^{i_0}$ and every inaccessible region V_{-i} is covered by exactly one robot. Hence proved.

Also, note that since the number of Voronoi cells is bounded by N (number of robots) and there is at least one Voronoi cell that is connected to any initially inaccessible region, therefore, the repartitioning technique takes at most N steps to find and connect the initially inaccessible region to another Voronoi cell. Consequently, the repartitioning mechanism is guaranteed to converge in a finite number of steps.

We have implemented the repartitioning algorithm using an auction protocol as shown in Algorithm 1. The robots use Voronoi partitioning to get their initial coverage regions corresponding to their Voronoi cells. Each robot then explores the boundary of its Voronoi cell. If, upon completing the exploration of its boundary, there are unexplored regions remaining in the Voronoi cell, these regions are allocated to neighboring robots using an auction protocol—robots in the neighboring Voronoi cells of the obstructed robot are sent a bid request message. Every neighbor robot calculates a bid for the region, and sends it to the auctioning robot. In the current implementation of the algorithm, these bids are calculated as the perimeter of the robot's current region. The robot that submits the lowest bid is selected as the winner of the auction and assigned the inaccessible portion of the Voronoi cell. The auctioning robot informs the winner, which then appends the region to the list of regions it needs to cover, and starts to perform boundary coverage of its newly assigned region. The auction algorithm possesses the essential properties (completion, non-overlapping coverage), but it reduces communication and coordination overhead by combining adjacent patches belonging to different robots, when the patches are accessible from each other.

Algorithm 1: Algorithm used by a robot to perform repartition coverage.

Repart-Coverage(V_i)
Input: V_i: Voronoi cell of robot i
Output: V_i': Repartitioned coverage region for robot
perform boundary coverage in V_i and determine $V_i^{i_0}$
$\mathbf{S}_{ij}^b \leftarrow$ set of blocked patches comprising $V_i \backslash V_i^i$
for *each* $S_{ij}^b \in \mathbf{S}_{ij}^b$ **do**
  $\mathbf{j} \leftarrow$ set of Voronoi neighbor robots of i that have Voronoi cell boundaries with S_{ij}^b
  send coordinates of polygon representing S_{ij}^b to all robot in $\mathbf{j}$
  wait for bids
  bid $\leftarrow$ set of bids received
  $j_{win} \leftarrow \arg\min_j$ **bid**
  $V_i \leftarrow V_i \backslash S_{ij}^b$ //remove S_{ij}^b from V_i
  send message to robot j_{win} to add S_{ij}^b to $V_{j_{win}}$
handleBidMessages() //for robot j
if *received bid request for* S_{ij}^b *from robot* i **then**
  $bid_j = \begin{cases} \text{currently covered perimeter of } V_j, & \text{if } S_{ij}^b \text{ reachable} \\ \infty, & \text{otherwise} \end{cases}$
  send bid_j to robot i
if *received winner message for* S_{ij}^b *from robot* i **then**
  $V_j \leftarrow V_j \cup S_{ij}^b$ //add S_{ij}^b to V_j
  Repart-Coverage(V_j)

5 Experimental Results

We have implemented our proposed Repart-Coverage algorithm using simulated e-puck robots within the Webots simulator. E-puck robots use a ring of eight IR-based proximity sensors with a 4 cm range to avoid obstacles and follow obstacle boundaries. Robots use Bluetooth protocol for inter-robot communication, and have a GPS and compass for localizing w.r.t the environment. Figure 6a–d, show four different environments measuring 2×2 m^2 with different internal obstacles and with 5–7 robots, placed initially at arbitrary positions. These environments illustrate different scenarios where the Voronoi cell of one or more robots becomes partially inaccessible due to the obstacles in the environment, corresponding to the different scenarios discussed in Sect. 4. The red lines on the floor of the environment denote the Voronoi cells assigned to each robot. For reaching and following the boundary of its Voronoi cell, each robot uses a lightweight, bug-like algorithm called Egress [13] that enables a robot to start from any arbitrary internal point in its assigned region, find a path to the region's boundary using basic motions such as move-outward and wall-follow, and, completely explore the entire outermost boundary of the region. Each robot's initial location is at the center of its Voronoi cell; the path followed by the robot is marked with a dark red trail. Figure 6e–h show the scenarios for the different

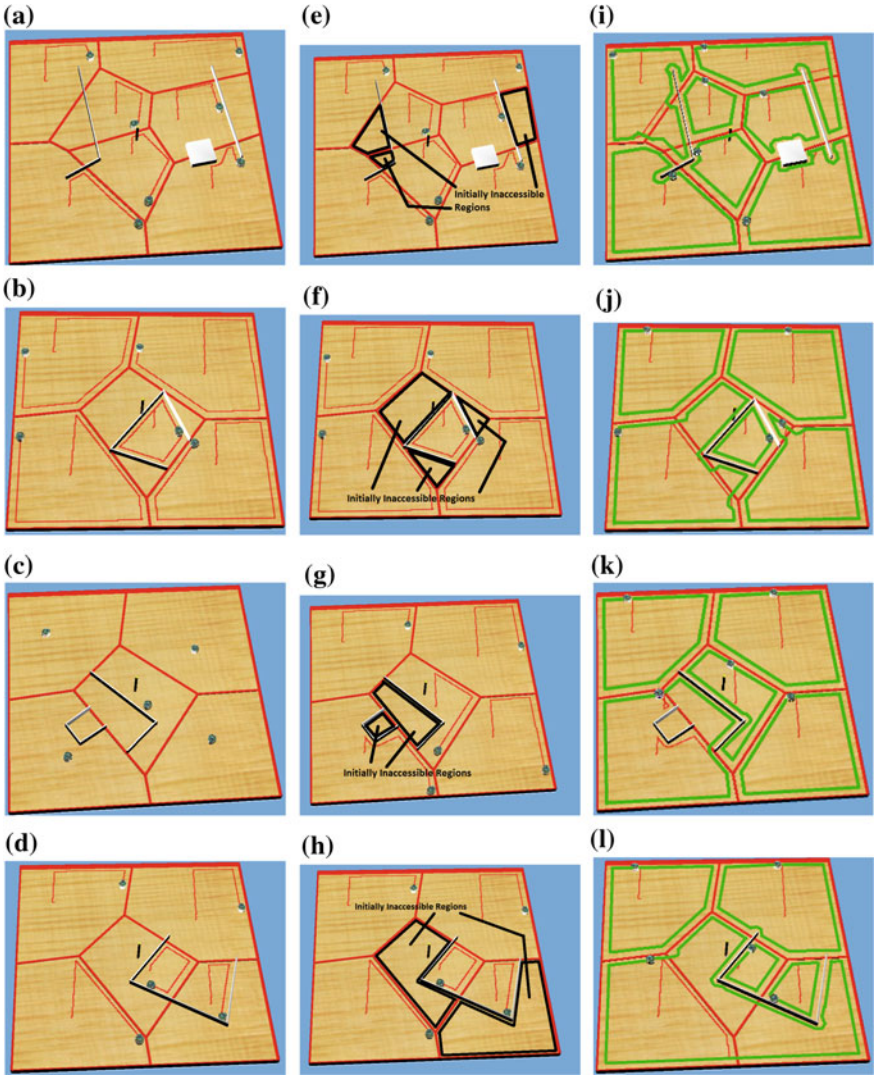


Fig. 6 Snapshots from Webots showing repartition coverage by 5–7 robots in different 2×2 m^2 environments with different obstacles. **a–d** initial Voronoi partition, **e–h** robots performing boundary coverage on original Voronoi cell, while showing inaccessible regions arising out of original Voronoi partition, **i–l** repartitioned cells and robots completing boundary coverage of entire environment; the final boundary of the cell that each robot covered is marked with a *green line*

environments at the end of boundary coverage along the Voronoi cell boundaries; the initially inacccesible regions of the respective Voronoi cells are marked with a black boundary. Finally, Fig. 6i–l show the result of our repartitioning algorithm. Robots from adjacent cells are allocated to cover each of the initially inaccessible regions

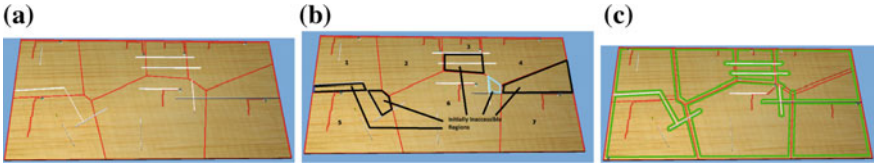


Fig. 7 Snapshots from Webots showing repartition coverage by 7 robots in a $3 \times 6\,m^2$ environment with different obstacle features, **a** initial Voronoi partition, **b** robots performing boundary coverage on Voronoi cell, *black/light blue boundaries* show inaccesible regions. **c** repartitioned cells and robots completing coverage of entire environment

using the Repart-Coverage algorithm. The trail of the paths followed by the different robots shows that every region in the environment is covered by exactly one robot. This shows that our algorithm is successful to (re)-parititon the free space in the environment into complete, non-overlapping regions for coverage.

Figure 7a–c show another instance of the operation of the Repart-Coverage algorithm for a $3 \times 6\,m^2$ environment with 7 robots. The scenario includes some unique obstacle features like narrow channels between obstacles and obstacles that span across multiple Voronoi cells, which require the inaccessible regions to be re-allocated to robots multiple times (similar to scenarios iv. and v. in Sect. 4). This shows that our algorithm successfully terminates and is able to find complete, non-overlapping regions even for complex obstacle geometries.

Finally, we have quantified the performance of our algorithm in terms of the area allocated to the different robots and the distances covered by them while performing boundary coverage. Table 1 shows the average area of the region allocated to each robot using our algorithm versus the area of the initial Voronoi cell for the different environments we have considered. Note that the initial Voronoi partition results in uncovered regions while the repartitioning guarantees complete coverage. The results for the different environments show that when obstacles result in larger inaccessible regions in the initial Voronoi cells, the coverage regions for each robot recalculated by the repartitioning algorithm have higher variance (std. dev, and max/min) than the initial Voronoi cells. This is because, with more complex obstacles, robots have to cover regions from other robots' initial Voronoi cells in addition to covering their own Voronoi cells.

6 Conclusions and Future Work

We proposed a novel technique for distributed spatial partitioning of an initially unknown region that guarantees a partitioning of the free space in the environment into a set of connected regions that can be covered by each robot. Currently, we are investigating techniques for each robot to dynamically build a map of the boundary of its currently allocated region instead of maintaining the end points of vertices of

Table 1 Area of cells to be covered before and after repartitioning for the different environments shown in Figs. 6a–d and 7a

Area of cell (m^2)	Env.9(a) before	Env.9(a) after	Env.9(b) before	Env.9(b) after	Env.9(c) before	Env.9 (c) after	Env.9(d) before	Env.9(d) after	Env.10(a) before	Env.10(a) after
Avg.	0.581	0.574	0.8	0.804	0.8	0.798	0.8	0.8	2.57	2.57
Stdev.	0.175	0.244	0.149	0.361	0.149	0.251	0.149	0.541	1.111	1.151
Min.	0.36	0.25	0.58	0.21	0.58	0.39	0.58	0.21	0.99	0.45
Max.	0.78	0.85	0.99	1.17	0.99	1.01	0.99	1.58	4.38	4.28
Uncov.	0.34	0	0.4	0	0.24	0	0.76	0	2.07	0

the boundary segments. The boundary map will enable a robot to efficiently plan its path to newly added regions instead of circumventing regions whose boundary it has already explored. Additionally, with boundary maps, the load (area covered) between different robots can be balanced by including factors such as the area of and distance to the newly allocated region, and, the area of the existing region in the robots' bids for new regions. Finally, we are implementing the proposed algorithm on physical robots.

Acknowledgments This work was partially supported by the U.S. Office of Naval Research as part of the COMRADES project.

References

1. Choset, H.: Coverage of known spaces: the boustrophedon cellular decomposition. Auton. Robots **9**, 247–253 (2000)
2. Hert, S., Lumelsky, V.: Polygon area decomposition for multiplerobot workspace division. Int. J. Comput. Geom. Appl. **8**, 437–466 (1998)
3. Rekleitis, I., New, A.P., Rankin, E.S., Choset, H.: Efficient boustrophedon multi-robot coverage: an algorithmic approach. Ann. Math Artif. Intell. **52**, 109–142 (2008)
4. Cortes, J., Martinez, S., Karata, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Rob. Auton. **20**(2), 243–255 (2004)
5. Bash, B.A., Desnoyers, P.J.: Exact distributed voronoi cell computation in sensor networks. In: Proceedings of the Sixth IEEE/ACM Conference On Information Processing in Sensor Networks, pp. 236–243 (2007)
6. Choset, H.: Coverage for robotics—a survey of recent results. Ann. Math. Artif. Intell. **31**, 113–126 (2001)
7. Altshuler, Y., Yanovski, V., Wagner, I.A., Bruckstein, A.M.: Multi-agent cooperative cleaning of expanding domains. I. J. Robot. Res. **30**(8), 1037–1071 (2011)
8. Agmon, N., Hazon, N., Kaminka, G.: The giving tree: constructing trees for efficient offline and online multi-robot coverage. Ann. Math Artif. Intell. **52**(2–4), 143–168 (2009)
9. Jäger, M., Nebel, B.: Dynamic decentralized area partitioning for cooperating cleaning robots. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3577–3582 (2002)
10. Schwager, M., Rus, D., Slotine, J.-J.E.: Decentralized, adaptive coverage control for networked robots. I. J. Robot. Res. **28**(3), 357–375 (2009)
11. Breitenmoser, A., Schwager, M., Metzger, J.-C., Siegwart, R., Rus, D.: Voronoi coverage of non-convex environments with a group of networked robots. In: ICRA, pp. 4982–4989 (2010)
12. Durham, J.W., Carli, R., Frasca, P., Bullo, F.: Discrete partitioning and coverage control for gossiping robots. IEEE Trans. Robot. **28**(2), 364–378 (2012)
13. Guruprasad, K.R., Dasgupta, P.: Egress: an online path planning algorithm for boundary exploration. In: IEEE International Conference on Robotics and Automation, May 2012, pp. 3991–3996
14. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math Artif. Intell. **31**, 77–98 (2001)
15. Batalin, M., Sukhatme, G.S.: Spreading out: a local approach to multi-robot coverage. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, Fukuoka, Japan, Jun 2002, pp. 373–382. [Online]. http://robotics.usc.edu/publications/56/

On Combining Multi-robot Coverage and Reciprocal Collision Avoidance

Andreas Breitenmoser and Alcherio Martinoli

Abstract Although robotic coverage and collision avoidance are active areas of robotics research, the avoidance of collision situations between robots has often been neglected in the context of multi-robot coverage tasks. In fact, for robots of physical size, collisions are likely to happen during deployment and coverage in densely packed multi-robot configurations. For this reason, we aim to motivate by this paper the combined use of multi-robot coverage and reciprocal collision avoidance. We present a taxonomy of collision scenarios in multi-robot coverage problems. In particular, coverage tasks with built-in heterogeneity such as multiple antagonistic objectives or robot constraints are shown to benefit from the combination. Based on our taxonomy, we evaluate four representative robotic use cases in simulation by combining the specific methods of Voronoi coverage and reciprocal velocity obstacles.

Keywords Multi-robot coverage · Voronoi tessellation · Reciprocal collision avoidance · Velocity obstacles · Taxonomy of collision scenarios · Evaluation of use cases

1 Introduction

The primary objective of multi-robot coverage involves the *deployment* and/or *sweeping motion* of a group of mobile robots within a region or along boundaries in order to provide a service, such as monitoring or maintenance. Whenever the coverage tasks require the robots to come close, higher-priority objectives of cooperation are imposed, including the avoidance of *robot-to-robot collisions*. Collision

A. Breitenmoser (✉)
Robotic Embedded Systems Laboratory, Department of Computer Science,
Viterbi School of Engineering, University of Southern California, Los Angeles, USA
e-mail: andreas.breitenmoser@usc.edu

A. Martinoli
Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture,
Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne,
Lausanne, Switzerland
e-mail: alcherio.martinoli@epfl.ch

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_4

situations arise first of all due to the other robots that are involved in the same task of covering the mission space. Second, collisions with other independent robots present in the mission space must be avoided. These robots—either static or dynamic—are pursuing their own objectives in a collaborative or non-collaborative fashion.

A survey of robotic coverage is given in [1]. Deployment and sweeping motion for robotic coverage of areas and boundaries (e.g., barriers) have previously also been referred to as blanket, barrier and sweep coverage [2]. A particular type of blanket coverage is *Voronoi coverage* [3], which arranges the robots in a final configuration that forms a so-called Centroidal Voronoi Tessellation (CVT) [4].

In the context of Voronoi coverage, robot-to-robot collision avoidance for robots of physical size (i.e., finite size instead of zero-sized point robots) has previously been considered by [5, 6]. The method in [5] restricts the robots' positions to the collision-free subareas in the interiors of their Voronoi cells; the Voronoi coverage controller in [6] adds a collision avoidance component based on repulsive terms to the coverage control law. Both methods, however, focus on one collision scenario only, which addresses the collision avoidance among robots that all share one single objective and execute the same Voronoi coverage control law cooperatively.

The contributions of this paper are threefold. First, we describe the possible types of robot-to-robot collision scenarios in multi-robot coverage problems and propose a taxonomy (Sect. 2). Second, we present a concrete solution for integrating a reciprocal collision avoidance algorithm into a multi-robot coverage algorithm; in particular, the CVT-based Voronoi coverage controller is combined with *Reciprocal Velocity Obstacles* (RVO), using the Optimal Reciprocal Collision Avoidance (ORCA) formulation (Sect. 3). Besides collision-free coverage, this allows for collision avoidance between heterogeneous robots (e.g., robots with different kinematic models). Third, we evaluate four use cases for combining multi-robot coverage and reciprocal collision avoidance, which show some characteristics that are inherent to such a combination (Sect. 4). Final conclusions are provided in Sect. 5.

2 A Taxonomy of Collision Scenarios in Multi-robot Coverage

In this paper, we deal with instances of multi-robot coverage problems, i.e., problems which ask for covering a mission space with multiple robots. Each robot has its own primary objective, which may be an *individual* or a *shared* common goal with other robots. The robots that share common goals are in the following considered members of the same *group* or *team*. The primary objective of at least a subset of the robots will be the coverage of the common mission space.

In such a setting, there are many possibilities for conflicting situations, so-called *collision scenarios*, which need to be resolved. Some scenarios are encountered during initial robot deployment and others in a later stage of the coverage process. Some scenarios occur among robots of the same team, i.e., *intragroup*, and others

between robots that belong to different teams, i.e., *intergroup*. In some collision scenarios, the robots must avoid each other while they collaborate, yet in others, the robots may compete and collisions with adversary robots must be avoided.

2.1 Categorization of Collision Scenarios

We base our categorization of collision scenarios in multi-robot coverage on the categorization of coverage behaviors by [2] and the categorization of interactions among agents by [7], where interactions are classified along the axes of "individual or shared goals", "actions advance goals of others", and "awareness of others". For the multi-robot coverage tasks of our interest, we assume that the robots are aware of each other. Consequently, our taxonomy has three dimensions:

- **Coverage phases during deployment and sweeping**: We distinguish between the two coverage types of deployment and sweeping motion, each of which is subdivided into two *coverage phases*. Deployment refers to blanket and barrier coverage: a robot team deploys in the first phase and assumes a static coverage configuration. In the second phase after the initial deployment, the robots observe the mission space from their configuration. Sweeping motion refers to sweep coverage (and coverage by a moving barrier): each robot covers the mission space by a sweeping motion in the first phase. If a second phase exists, the robots move over already covered space and relocate, inspect a covered location closer, or resume the sweeping motion to achieve persistent or redundant coverage of the mission space.
- **Intragroup and intergroup collision avoidance**: According to [7], robots may share common goals or have individual differing goals. Robots with shared common goals form a team or group. Single robots or robots of different teams are said to be *external* to each other. During the completion of a task, such as coverage, the robots must avoid collisions and resolve collision situations inside their own team (intragroup) as well as between external robots and teams (intergroup). The robot teams may be homogeneous or as well consist of heterogeneous robots with different sizes, sensing and mobility capabilities (e.g., different kinematics).
- **Cooperative and non-cooperative behavior**: For coverage and reciprocal collision avoidance, the degree of *cooperation* is another important factor. Similar to [7], we measure cooperativeness by whether the actions of one robot influences the goals of other robots (both individual or shared goals) in a positive or negative way. Positive influence represents cooperative behaviors, neutral or negative influence represents non-cooperative, including competing or adversary, behaviors. Non-cooperative robots appear to each other as *static or dynamic obstacles*.

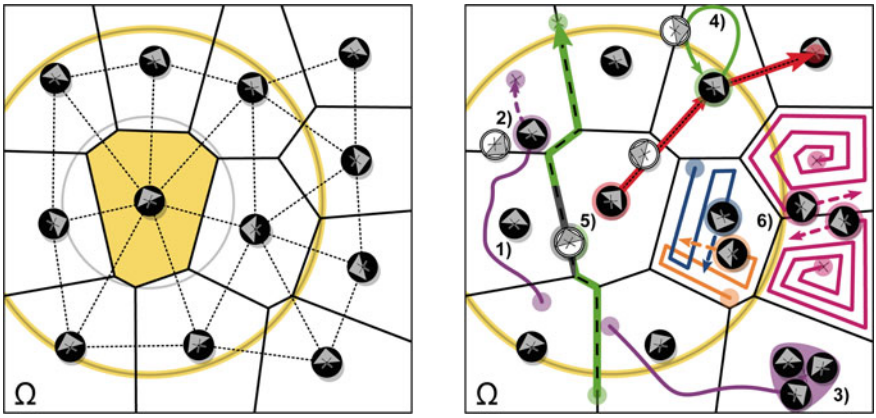


Fig. 1 Coverage without (*left*) and with (*right*) reciprocal collision avoidance. *Left* A robot team (*black* robots) performs Voronoi coverage and covers a mission space Ω by creating a CVT. The Voronoi graph is formed by the boundaries of the Voronoi cells (full *black lines*) and the dual Delaunay graph is visualized by the *black dashed lines*. The Voronoi neighborhood of one of the robots (*yellow* Voronoi cell) is indicated by the *yellow outer circle*. *Right* The robot team now avoids collisions among each other and with external robots (*white* robots); the numbers (1)–(3) and (4)–(6) refer to the different collision scenarios that occur during (i.e., first coverage phase) and after (i.e., second coverage phase) the initial deployment (see Sect. 2)

2.2 The Collision Scenarios in Voronoi Coverage

The multi-robot Voronoi coverage control approach serves us as a demonstration example to show the different categories suggested by our taxonomy. The basic CVT-based Voronoi coverage controller[1] is an example of the deployment coverage type. In addition, we will also consider a hybrid variant, where the second coverage phase involves, instead of observing, sweeping motions in the Voronoi cells.

The collision scenario (1) in Fig. 1 on the right depicts the trajectory of a robot that shows intragroup collision avoidance and cooperative behavior when avoiding another team member (black robot) during the first phase of deployment. A similar situation is illustrated by collision scenario (2) in Fig. 1 but for intergroup collision avoidance between a team member that performs Voronoi coverage (black robot) and an external robot with differing goal (white robot).

Some collision scenarios include components from both intragroup and intergroup collision avoidance. Instead of single robots, a team of several robots—considered as a single entity—can as a subgroup itself be part of a larger team and thus be subject to coverage and collision avoidance. During the initial phase of deployment, the robots in the subgroup must avoid reciprocal collisions among each other locally and with other members of the team (intragroup collision avoidance, cooperative behavior), as well as with potential external robots (intergroup collision avoidance, cooperative or

[1] Refer to Fig. 1 on the left for an illustration and to Sect. 3 for a formal description.

non-cooperative). The collision scenario (3) in Fig. 1 shows such an abstraction for a team of three robots, which forms a subgroup of the overall covering team (black robots).

In the second coverage phase of the deployment, certain operations, such as recharging, servicing, escape or evasion maneuvers, require robots of a team to occasionally leave their positions in the static coverage configuration for a short time. During these operations, the robots must avoid reciprocal collisions with team members (intragroup or intergroup collision avoidance) as well as with approaching external robots (intergroup collision avoidance). The robots which undergo such escape and return maneuvers become in some cases instances of "on-off" team members, i.e., they are recognized as external robots by some or all of the team members for a limited time period. In other words, these robots temporally convert into external robots, and apply intergroup collision avoidance, but eventually rejoin the robot team. The collision scenario (4) in Fig. 1 gives an example of a robot that leaves its Voronoi cell and comes back after having moved to the mission space boundary. On its way, it may get involved into collisions, e.g., with a former team member or an external dynamic obstacle (white robot).

The initial deployment of a robot team often goes along with a decomposition or tessellation of the mission space. This provides an additional representation of the environment and robot configuration that can be shared among the robots. With respect to Voronoi coverage, the constructed CVT includes Voronoi and Delaunay graphs, which can be used as roadmaps for robot navigation in the mission space. Single robots or groups of multiple robots may, for example, patrol the Delaunay graph or pass threats with a maximum clearance or safety distance by transitioning the Voronoi graph. On both roadmaps, there can potentially be oncoming traffic of cooperative or non-cooperative robots, which asks for reciprocal intragroup or intergroup collision avoidance. The collision scenarios (5) in Fig. 1 show a robot (black) that moves along a path (red) on the Delaunay graph to a next Voronoi cell and an external dynamic obstacle (white robot) that moves along a path (green) on the Voronoi graph amidst the deployed robots. The first robot must actively avoid collisions on its path whereas the second robot needs to be avoided by other robots.

The two coverage types can also be combined; such hybrid coverage methods involve the hierarchical coupling of deployment and sweeping motion [8]. In case of Voronoi coverage, after the robots have deployed and a CVT spans the mission space, each robot in the team covers its Voronoi cell by sweeping motions (e.g., spiraling or back-and-forth sweeping patterns) during the second coverage phase. Here, collision situations occur during sweeping. The collision scenarios (6) in Fig. 1 illustrate that the robots must either avoid reciprocal collisions at the boundaries of their Voronoi cells (intragroup collision avoidance)[2] or within the cells, in case several robots—possibly of different teams (this would mean intergroup collision avoidance)—sweep the same Voronoi cell for purposes of redundant coverage.

[2]In real-world scenarios, with positional noise and varying pose estimates for each robot (different from Assumption 3 in Sect. 3), the resulting degenerate Voronoi cells may overlap, which naturally leads to collision situations even farther away from the boundaries of the Voronoi cells.

Intragroup and intergroup collision avoidance during the second coverage phase can involve cooperative or non-cooperative behavior. Whereas two robot team members would typically cooperate when facing a collision situation while driving along a sweeping path or path in a roadmap, the cooperativeness of a robot that temporally leaves its team, for example for recharging, may strongly depend on its current state, e.g., its urge due to a low remaining battery level. Moreover, external dynamic obstacles pursue their own goals by strictly acting in a non-cooperative way.

3 Combining Voronoi Coverage and RVO

We are now going to present the implementation of a concrete solution for reciprocal collision avoidance in a multi-robot coverage problem. We build on the aforementioned example of CVT-based Voronoi coverage and combine it with reciprocal collision avoidance in velocity space, using the RVO and ORCA[3] methods [9–12].

Voronoi coverage, as presented in Sect. 3.1, is based on a gradient-descent control law. In a collision situation, each involved robot faces one or multiple other robots in the mission space. Independent of their cooperativeness, each robot represents a dynamic (or static for stationary robots) obstacle that needs to be avoided completely, or up to a certain degree. This introduces (dynamically changing) constraints on the robot controllers and leads to constrained (or projected) gradient descent, which makes the problem considerably more challenging. We approach this *constrained optimization problem* with the reciprocal collision avoidance method using RVO and ORCA in Sect. 3.2.

3.1 CVT-based Voronoi Coverage

We restate the most important formulations from Voronoi coverage control after [3]. Given n robots at positions $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, which are tasked with covering the mission space $\Omega \subset \mathbb{R}^N$, let the coverage objective function be $\mathcal{H}_\mathcal{V}$ and the corresponding *coverage cost*

$$\mathcal{H}_\mathcal{V}(P) = \sum_{i=1}^{n} h(\mathbf{p}_i, V_i) = \sum_{i=1}^{n} \int_{V_i} f(d(\mathbf{q}, \mathbf{p}_i))\, \rho(\mathbf{q})\, dF(\mathbf{q}) \,. \tag{1}$$

The Voronoi tessellation over Ω is given by the set of Voronoi cells $\mathcal{V}(P) = \{V_1, \ldots, V_n\}$, where

$$V_i = \left\{ \mathbf{q} \in \Omega \mid d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j),\ j \neq i \right\} ,$$

[3] Under the linear programming formulation, the RVO method becomes the ORCA method.

$\forall\, i,\ j \in \{1, \ldots, n\}$. Two robots i and j are said to be *Voronoi neighbors* if their Voronoi regions V_i and V_j are adjacent. The density function $\rho\colon\ \Omega \to \mathbb{R}_{\geq 0}$ directs the robots to areas of special interest. The function to measure distance between locations $\mathbf{q} \in V_i$ and robot positions $\mathbf{p}_i$ is defined as $d\colon\ \Omega^2 \to \mathbb{R}_{\geq 0}$. The performance function $f\colon\ \mathbb{R}_{\geq 0} \to \mathbb{R}$, which must be strictly increasing over the image of d, measures the degradation of the coverage performance with increasing distance.

As further shown in [3–5], the objective function is minimized by solving

$$\begin{aligned} \nabla_{\mathbf{p}_i} \mathcal{H}_{\mathcal{V}}(P) &= \nabla_{\mathbf{p}_i} h(\mathbf{p}_i,\ V_i) \\ &= \int_{V_i} \nabla_{\mathbf{p}_i} f(d(\mathbf{q},\ \mathbf{p}_i))\ \rho(\mathbf{q})\ dF(\mathbf{q}) = \mathbf{0}\,. \end{aligned}$$

The partial derivatives and linear proportional control laws can then be obtained for each robot,

$$\begin{aligned} \nabla_{\mathbf{p}_i} h(\mathbf{p}_i,\ V_i) &= -2\ M_{V_i}\ (\mathbf{c}_{V_i} - \mathbf{p}_i)\,, \\ \mathbf{v}_i^{\text{pref}} &= -k\ \nabla_{\mathbf{p}_i} h(\mathbf{p}_i,\ V_i)\,, \end{aligned} \tag{2}$$

with M_{V_i} and k set to positive values. The centroids $\mathbf{c}_{V_i}$ are critical points of the objective function $\mathcal{H}_{\mathcal{V}}$. The *preferred velocities* $\mathbf{v}_i^{\text{pref}}$, which are tracked by the robots, point toward the centroids and make the robots iteratively approach the centroids; the resulting CVT at convergence leads to a local minimum of the objective function. Figure 1 on the left shows a CVT, the Voronoi graph and its dual, the Delaunay graph, as well as an example of a robot's Voronoi neighborhood.

We will additionally make the following assumptions with respect to the Voronoi coverage control in this paper.

Assumption 1 The Voronoi tessellation is defined after [3] by a coverage objective function that consists of the Euclidean distance $d(\mathbf{q},\ \mathbf{p}_i) = \|\mathbf{q} - \mathbf{p}_i\|_2$ and the performance function $f(d(\cdot)) = d(\cdot)^2$. Under these settings, M_{V_i} is the mass of the Voronoi cell V_i for a given area density function ρ; we assume constant density $\rho(\cdot) = 1$.

Assumption 2 We assume the mission space Ω to be two-dimensional, i.e., $N = 2$, and convex. In particular, the mission space does not contain any static obstacles as fixed components of the environment. However, there are mobile robots in the mission space, which represent—depending on whether they are moving or stopped—dynamic and static obstacles of circular shape to one another (see Fig. 1).

Assumption 3 The robots sense noisy positions; the noise in a robot's position is uniformly distributed over a circle centered at the noise-free actual position and its radius is bounded by a maximum noise amplitude. We assume one noisy position per robot, i.e., each robot's own position estimate and the estimates of its position by the other robots are equivalent. Due to this assumption, the Voronoi tessellations are correct partitions of Ω, composed of fully covering, disjoint sets of Voronoi cells.

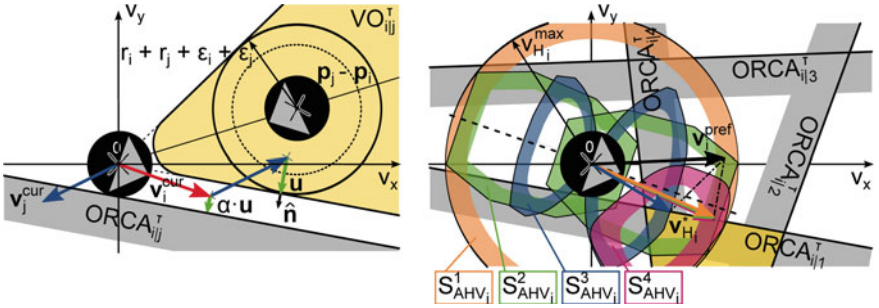


Fig. 2 RVO and ORCA. *Left* Construction of the set of collision-free velocities $ORCA^{\tau}_{i|j}$ from $VO^{\tau}_{i|j}$ for robot i. *Right* Computation of the optimal holonomic velocities $\mathbf{v}^{*}_{H_i}$ from the set $ORCA^{\tau}_{i}$, given four other robots in robot i's neighborhood as well as four different possible sets of allowed holonomic velocities S_{AHV_i}. $S^1_{AHV_i}$ represents the set for a holonomic robot, $S^2_{AHV_i}$ corresponds to a differential-drive robot, and $S^3_{AHV_i}$ and $S^4_{AHV_i}$ describe two instances of the set for a bicycle robot

3.2 Reciprocal Collision Avoidance Using RVO and ORCA

Assumptions 2 and 3 likewise hold for the reciprocal collision avoidance. In the case of holonomic robots with velocities $\mathbf{v}_H$, any robot j with radius r_j positioned at $\mathbf{p}_j$, within a given neighborhood[4] of a robot i with radius r_i positioned at $\mathbf{p}_i$ and $j \neq i$, induces a *velocity obstacle*

$$VO^{\tau}_{i|j} = \left\{ \bar{\mathbf{v}} \mid \exists t \in [0\,,\ \tau]\,,\ t \cdot \bar{\mathbf{v}} \in D(\mathbf{p}_j - \mathbf{p}_i\,,\ r_i + r_j) \right\}\,. \tag{3}$$

The vectors $\bar{\mathbf{v}} = \mathbf{v}_{H_i} - \mathbf{v}_{H_j}$ form the set of relative velocities between the robots, τ is the time horizon for a collision to occur and $D(\mathbf{p}\,,\ r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\|_2 < r\}$ is the open ball of radius r. The RVO and ORCA methods [9, 10] now assume that all the robots make similar attempts in order to avoid collisions. The *set of collision-free velocities* $ORCA^{\tau}_{i|j}$ for a robot i with respect to any other robot j in its neighborhood results from $VO^{\tau}_{i|j}$ through an adjustment in velocity by

$$\mathbf{u} = \underset{\bar{\mathbf{v}} \in \partial VO^{\tau}_{i|j}}{\operatorname{argmin}} (\|\bar{\mathbf{v}} - (\mathbf{v}^{\mathrm{cur}}_i - \mathbf{v}^{\mathrm{cur}}_j)\|_2) - (\mathbf{v}^{\mathrm{cur}}_i - \mathbf{v}^{\mathrm{cur}}_j)\,.$$

The vector $\mathbf{u}$ represents the smallest change the robot needs to add to the difference in the current velocities of the robots, $\mathbf{v}^{\mathrm{cur}}_i - \mathbf{v}^{\mathrm{cur}}_j$, in order to fully avoid a collision. The *cooperativeness ratio* $\alpha \in [0\,,\ 1]$ scales $\mathbf{u}$ and defines to which extent a single robot eventually participates in avoiding a reciprocal collision. The construction of the set $ORCA^{\tau}_{i|j}$ is shown on the left of Fig. 2. $ORCA^{\tau}_{i|j}$ is given as

$$ORCA^{\tau}_{i|j} = \{\mathbf{v}_{H_i} \mid (\mathbf{v}_{H_i} - (\mathbf{v}^{\mathrm{cur}}_i + \alpha \cdot \mathbf{u})) \cdot \hat{\mathbf{n}} \geq 0\}\,, \tag{4}$$

[4] In our case, the Voronoi neighborhood can be used as neighborhood in the RVO computation.

where $\hat{\mathbf{n}}$ denotes the outward normal at $(\mathbf{v}_i^{\text{cur}} - \mathbf{v}_j^{\text{cur}}) + \mathbf{u}$ on $\partial\, VO_{i|j}^{\tau}$, i.e., the boundary of $VO_{i|j}^{\tau}$. Let S_{AHV_i} be the *set of allowed holonomic velocities* given the kinematic constraints of robot i. The final set of collision-free velocities is then computed as

$$ORCA_i^{\tau} = S_{AHV_i} \cap \bigcap_{j \neq i} ORCA_{i|j}^{\tau} \,. \tag{5}$$

The right side of Fig. 2 illustrates the set $ORCA_i^{\tau}$ in a multi-robot scenario for different types of S_{AHV_i}, including $S_{AHV_i} = D(0\,,\, v_{H_i}^{max})$ for holonomic robots with an upper bound on the velocity of $v_{H_i}^{max}$, as well as the S_{AHV_i} for differential-drive and bicycle (respectively car-like) robots, whose detailed derivations can be found in [11, 12]. The extension of the ORCA method to robots with non-holonomic kinematics is based on the idea that a robot i with given kinematic constraints can be enabled by a trajectory tracking controller to track a set of allowed holonomic velocities S_{AHV_i} within a certain maximum error bound. Because of the enlargement of the robots' radii by this bound, $r_i' = r_i + \varepsilon_i$ and $r_j' = r_j + \varepsilon_j$, the robots can be treated as if holonomic. The velocity obstacles $VO_{i|j}^{\tau}$ in (3), the set $ORCA_{i|j}^{\tau}$ in (4) and as a result the set $ORCA_i^{\tau}$ in (5) are modified by the extended radii r_i' and r_j' in this case. This offers the flexibility of forming heterogeneous groups of multiple robots with different kinematic constraints and using them together in a common coverage task.

We finally obtain the optimal holonomic velocity of robot i by projection to $ORCA_i^{\tau}$,

$$\mathbf{v}_{H_i}^{*} = \underset{\mathbf{v}_{H_i} \in ORCA_i^{\tau}}{\text{argmin}} \left(\|\mathbf{v}_{H_i} - \mathbf{v}_i^{\text{pref}}\|_2 \right), \tag{6}$$

which avoids reciprocal collisions among all the robots in the neighborhood for at least the time horizon τ but also lies as close as possible to the previously specified preferred velocity $\mathbf{v}_i^{\text{pref}}$ of (2), which represents the primary objective of coverage.

Concerning the RVO and ORCA computation, the following additional assumption applies for the rest of the paper.

Assumption 4 Even though various cooperativeness ratios are supported, we set a robot's ratio to $\alpha = 0.5$ with respect to other cooperative robots, i.e., the robots avoid collisions in equal parts (cooperative behavior), and to $\alpha = 1$ with respect to other non-cooperative robots (non-cooperative behavior); these robots represent dynamic obstacles, which have to be fully avoided.

3.3 Properties of the Combined Method

When combining Voronoi coverage and RVO, we have two alternatives to compose distributed controllers. If we apply Voronoi coverage control in an outer loop at high level, the preferred velocities $\mathbf{v}_i^{\text{pref}}$ after (2) serve as inputs to the inner control loop given by the ORCA method in (6). This is the implementation we will use

throughout Sect. 4. Alternatively, Voronoi coverage can be used as inner loop for formation control of a group similar to [3, 13][5] and the entire group can be guided as single entity similar to [14, 15] by RVO or ORCA in the outer control loop.

Regarding the combined method in light of the taxonomy, "awareness of others" is implied by the Voronoi tessellation and the velocity obstacles of RVO. The CVT-based controller realizes the shared goal of well-balanced coverage and the cooperation behavior is expressed by α. The two coverage phases are given by the two periods before and after convergence of the Voronoi coverage controller.

The unconstrained Voronoi coverage controller is shown in [3] to converge for a team of cooperatively covering holonomic robots. In the constrained case (Voronoi coverage combined with RVO), a team of cooperatively covering holonomic robots with intragroup collision avoidance converges but stays off the centroids $\mathbf{c}_{V_i}$ by $(r_i' + r_j')/2$ in the worst case. A team of cooperatively covering non-holonomic robots converges after [3, 6] whenever the robots move closer to the centroids in each control step (in particular, this requires bidirectional driveability of the robots). Intergroup collision avoidance among non-cooperative robots or teams, however, can introduce arbitrary perturbances, such that final convergence is not guaranteed.

4 Simulation of Robotic Use Cases

In this section, we finally apply the combined Voronoi coverage and reciprocal collision avoidance methods from Sect. 3 in simulation,[6] and evaluate the collision scenarios from Sect. 2 for four representative robotic use cases.

4.1 Recharging Use Case

Our first use case of recharging relates to the collision scenarios (1), (2) and (4) in Fig. 1: four robots use the combined Voronoi coverage and reciprocal collision avoidance methods to cover a square mission space; during the process of coverage, the robots regularly run out of power and need to recharge their batteries. A Voronoi coverage-based method was applied to a similar application in [16], thereby focusing on energy-awareness regardless of collision situations.

The four robots are modeled after the Khepera III robots[7] as differential-drive robots with identical parameters; only their initial battery levels differ from each other. The robots can recharge by driving to the lower borderline of the mission space, which is the charging area (blue region in Fig. 3a–d). The robots start in the bottom left corner and are deployed in the mission space of $1.2 \times 1.2\,\mathrm{m}^2$. Energy

[5] Intragroup collision avoidance, however, is not considered in [3, 13].

[6] All the simulations have been conducted in the Matlab environment.

[7] See http://www.k-team.com/mobile-robotics-products/khepera-iii.

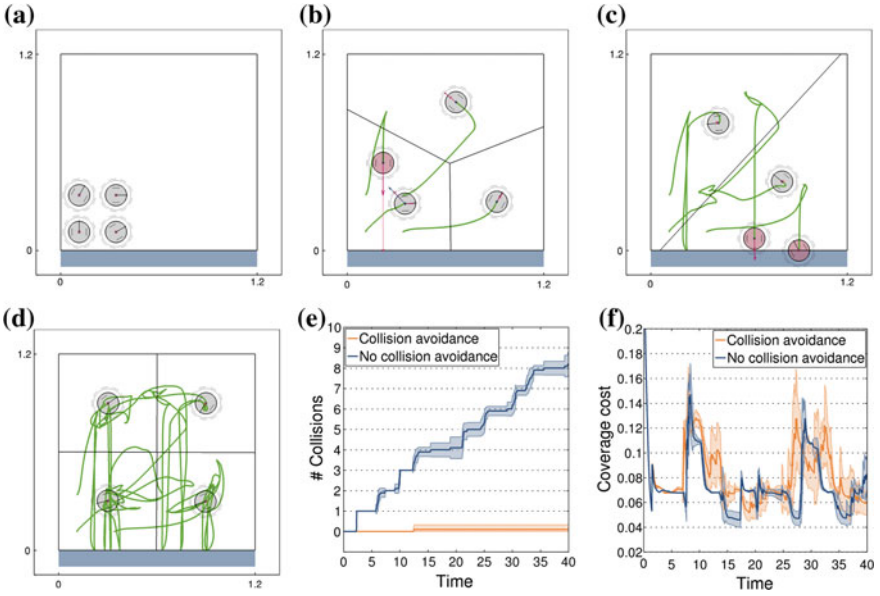


Fig. 3 Recharging use case. Four robots deploy from the *bottom left* corner of the mission space. During the coverage process, the robots leave the robot team, recharge at the lower borderline and rejoin the team after charging. The covering robots compensate for the recharging robots while concurrently avoiding collisions with them. No collision avoidance leads to faster convergence of the coverage cost but also to collisions. The envelopes show 95 % confidence intervals on the mean

is consumed per distance traveled and per time a robot is sensing. Sensing happens whenever a robot applies the Voronoi coverage controller, i.e., the robot is not moving to, returning from or residing at the charging area.

The recharging robots are examples of "on-off" team members, during as well as after the first phase of deployment. During deployment and after convergence, as soon as a robot's battery level decreases below a minimum critical value (red robots in Fig. 3b–c), the robot leaves the team of covering robots and becomes an external dynamic obstacle to them (transition from intragroup to intergroup collision avoidance). The recharging robot will moreover become non-cooperative and will not help to avoid collisions with the covering robots anymore. This can also be viewed as inherently increasing the priority of the recharging robot, i.e., the remaining covering robots now have to give way and fully avoid collisions with that robot. However, the covering robots as well as the recharging robots themselves remain cooperative and still avoid collisions among each other in equal parts. Once fully charged, the robots return to the last position at which they were located when the critical battery level was detected, and rejoin the covering robot team.

Figure 3a–d presents several snapshots from the simulation of the recharging scenario. The noise in the robots' positions is bounded by a maximum value of 0.01 m. We compare the combined Voronoi coverage and reciprocal collision avoidance

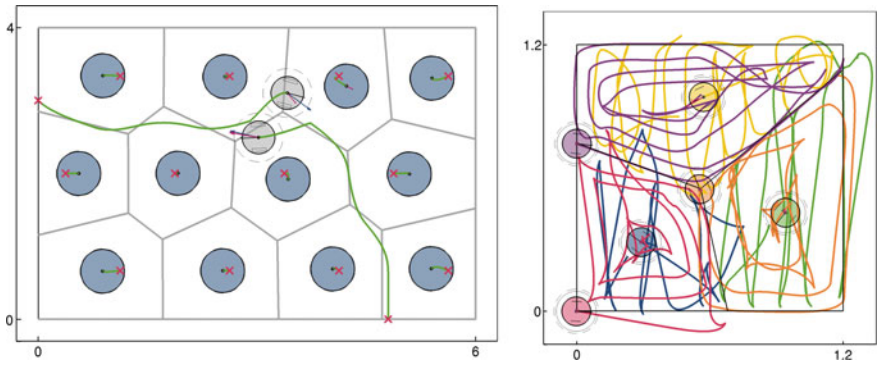


Fig. 4 Push-through (*left*) and sweeping (*right*) use cases. *Left* Two robots (*gray*) push through human agents (*blue*) in a $4 \times 6\,\mathrm{m}^2$ mission space. The *red* points mark the start positions, the *gray lines* the Voronoi cells and the trajectories are in *green*. Thin *blue arrows* represent velocities $\mathbf{v}_i^{\mathrm{pref}}$ and *red arrows* indicate velocities $\mathbf{v}_{H_i}^*$. *Right* Two robots at a time form a group (*red* and *blue*, *violet* and *yellow*, *orange* and *green*) and deploy. Each of the three groups covers one of the three Voronoi cells redundantly; the first robot in the group (*red*, *violet*, *orange*) executes a spiraling sweeping pattern and the second robot moves back and forth (*blue*, *yellow*, *green*). In the process, the robots have to avoid reciprocal collisions at the boundaries of their Voronoi cells and with each other. Finally, the set of *red*, *violet* and *orange* trajectories and the set of *blue*, *yellow* and *green* trajectories each result in complete coverage of the mission space without collisions

methods with the case where no collision avoidance is performed. Each case is tested by 10 simulation runs, during which each robot recharges twice in average. Without collision avoidance, the coverage cost is minimized faster and reaches lower levels (Fig. 3f). The resulting configurations are more optimal in terms of the minimization of the coverage cost in (1) since the covering robots do not need to avoid the recharging robots. However, there occur an average total of 8 collisions during each simulation run and 80 collisions over all runs, whereas the use of the collision avoidance method prevents most of these collisions[8] (Fig. 3e).

4.2 Push-Through Use Case

In the second use case, we simulate a heterogeneous crowd of 12 human agents and two robots (see Fig. 4 on the left). We model the robots as the differential-drive Pioneer 3-DX[9] and assume a holonomic kinematic model for the humans. The CVT is used as a simplified model of the human personal space. The human agents are distributed according to the Voronoi coverage controller. This is an example of collision scenario (5) in Fig. 1, after initial deployment, with intergroup collision

[8] Only a single collision occurred in a situation where a covering robot was jammed in between two non-cooperative robots that moved in opposite directions from and to the charging area.

[9] See http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx.

avoidance and cooperative behavior. After convergence, the human agents stand at their positions slightly apart, similar to people waiting at a bus stop. The robots move across the mission space by pushing through the crowd; in order to reduce disturbances of the humans, the robots follow the Voronoi graph, which represents a maximum clearance roadmap. Thereby, both the human agents and the robots run the reciprocal collision avoidance method at the low level with $\alpha = 0.5$ for everyone.

4.3 Sweeping Use Case

The third use case relates to sweep coverage, e.g., for cleaning or inspection tasks, and includes collision scenarios that occur during and after the first phase of deployment, with intragroup collision avoidance and cooperative behavior. It showcases the concepts of abstractions for robot groups and hybrid coverage [8], illustrated as collision scenarios (3) and (6) in Fig. 1 above. Six differential-drive Khepera III robots form a covering team but further subdivide into groups of two. The groups deploy in the mission space of $1.2 \times 1.2\,\mathrm{m}^2$. At convergence, the final CVT is fixed and each of the three groups subsequently sweeps its Voronoi cell collaboratively by applying spiraling and back-and-forth sweeping patterns (see Fig. 4 on the right). All the robots run the reciprocal collision avoidance method with $\alpha = 0.5$. The resultant redundant coverage with two different coverage patterns in parallel presents a characteristic outcome of combining robotic coverage and reciprocal collision avoidance.

4.4 Perturbation Use Case

The last use case looks at the perturbation that is introduced into a multi-robot system through external dynamic obstacles. The dynamic obstacles traverse a bounded mission space, which is covered by a robot team according to the Voronoi coverage control law. The covering robots need to fully avoid the dynamic obstacles as well as the enclosing borderlines of the mission space.

This use case shows a scenario for the second phase after initial deployment, with intergroup collision avoidance and non-cooperative behavior, and is of general interest for applications with adversarial pursuers or intruders. However, in this paper, we are particularly interested in the aspect of how the inherent perturbation by dynamic obstacles influences the coverage cost and the optimality of the robot deployment. The configurations after convergence of the Voronoi coverage method correspond to local minima of the coverage cost. More optimal configurations can be reached through the perturbation of the robot team. This may also help to break off saddle points and symmetry configurations which sometimes result from CVTs [4].

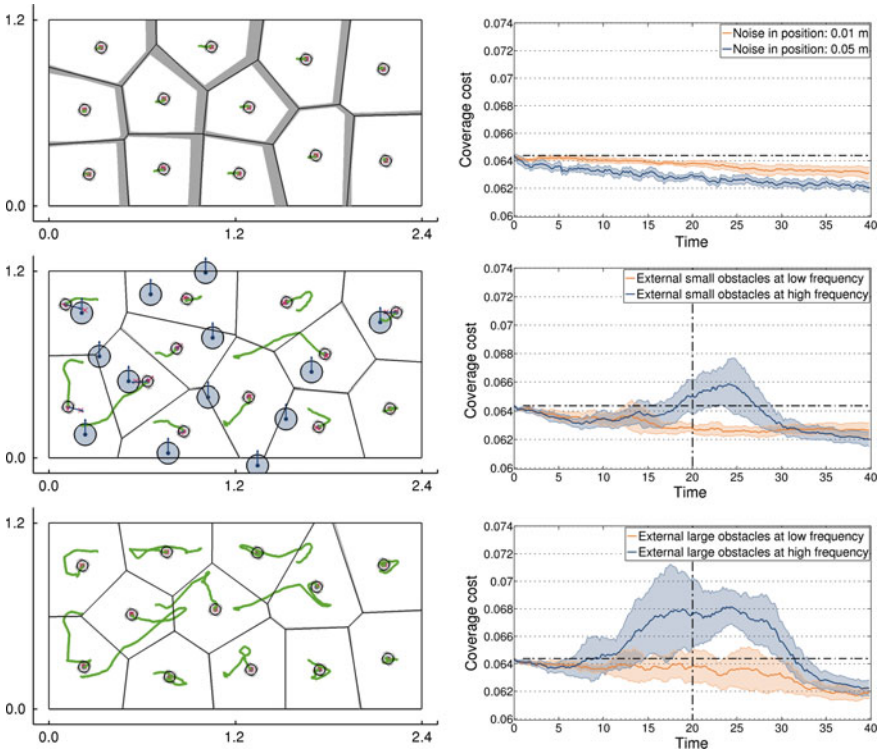


Fig. 5 Perturbation use case. *Top left* Start positions of the robots. The *gray lines* show how much the CVT is perturbed through noise for case (i) over a simulation run. *Center left* Massive perturbation of the covering robots by external dynamic obstacles (*blue*), moving from the *bottom* to the *top*, for case (vi). The robots avoid the collisions while covering the mission space. *Bottom left* Robots' final configuration after a completed simulation run for case (vi). *Top right* Coverage cost per time for cases (i) and (ii). *Center right* Coverage cost per time for cases (iii) and (iv). *Bottom right* Coverage cost per time for cases (v) and (vi). The envelopes show 95 % confidence intervals on the mean. The *black dash-dotted horizontal lines* show the cost at start and the *black dash-dotted vertical lines* at 20 s mark the time when the injection of obstacles is stopped

12 holonomic robots, similar in size to e-puck robots,[10] deploy initially from the bottom left corner. We simulate the cases with no perturbation through external dynamic obstacles but with a maximum noise in the robots' positions of (i) 0.01 m and (ii) 0.05 m, as well as the cases with perturbation for the maximum noise in position of 0.01 m and the following settings: (iii) small obstacles and low frequency of perturbation, (iv) small obstacles and high frequency of perturbation, (v) large obstacles and low frequency of perturbation, and (vi) large obstacles and high frequency of perturbation. The small obstacles have the same size as the robots, the large obstacles are double the size; the high frequency ($1\,s^{-1}$) is twice the low fre-

[10] See http://www.e-puck.org.

quency ($0.5\,s^{-1}$). The injection of new obstacles is stopped in each case after half the simulation time (20 s) to allow the robots to settle down. For each setting, 15 simulation runs were computed. Figure 5 shows the simulation of the perturbation scenario and compares perturbations through noise only with perturbations through small or large dynamic obstacles at low or high frequency. At low noise levels of 0.01 m, the robot configurations are not changed substantially. However, at increased noise levels, such as 0.05 m, the noise influences the robots' positions and changes the configuration, which leads to more optimal coverage cost. The same result can be achieved through the perturbation with external dynamic obstacles. The obstacles initiate high temporary perturbations which may stop at some point, whereas the noise level usually remains. Note that large obstacles and high frequencies introduce stronger perturbations, which take longer to settle down but increase chances for reaching more optimal configurations and lower coverage cost.

5 Conclusions and Future Work

This paper motivates the combined use of coverage and collision avoidance methods for multi-robot systems. We present a taxonomy of collision scenarios in multi-robot coverage problems and illustrate the performance of the combined methods in simulations. For our specific study, we review the Voronoi coverage control and reciprocal collision avoidance methods, such as RVO and ORCA, and combine and apply them to four representative robotic use cases, namely recharging during persistent coverage, pushing through a human crowd, sweeping for inspection and reacting to perturbations introduced by external dynamic obstacles.

As direct continuation of the presented work, the combined Voronoi coverage and reciprocal collision avoidance methods are to be tested for each use case on the real robot platforms. The study of Voronoi coverage control under the influence of actuator and sensor noise presents another related research direction. Foremost, it would be interesting to study further coverage and cooperation tasks in view of the proposed taxonomy of collision scenarios.

Acknowledgments The work presented in this paper has been carried out at the Distributed Intelligent Systems and Algorithms Laboratory at EPFL. The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007–2013—Challenge 2—Cognitive Systems, Interaction, Robotics—under grant agreement No 601033—MOnarCH.

References

1. Choset, H.: Coverage for robotics—a survey of recent results. Ann. Math. Artif. Intell. **31**, 113–126 (2001)
2. Gage, D.W.: Command control for many-robot systems. In Proceedings of the Annual AUVS Technical Symposium, vol. 10, pp. 28–34 (1992)

3. Cortés, J., Martínez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
4. Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: applications and algorithms. SIAM Rev. **41**(4), 637–676 (1999)
5. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In Proceedings of the IEEE Conference on Decision and Control, pp. 3947–3952 (2008)
6. Dirafzoon, A., Menhaj, M.B., Afshar, A.: Decentralized coverage control for multi-agent systems with nonlinear dynamics. IEICE Trans. **94–D**(1), 3–10 (2011)
7. Parker, L.E.: Distributed intelligence: overview of the field and its application in multi-robot systems. J. Phys. Agents **2**(2), 5–14 (2008)
8. Breitenmoser, A.: Multi-robot coverage and path planning for the inspection of curved surfaces, Ph.D. dissertation, no. 21009, ETH Zurich (2013)
9. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation, In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1928–1935 (2008)
10. van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: Proceedings of the 14th International Symposium on Robotics Research, STAR, vol. 70, pp. 3–19 (2011)
11. Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Proceedings of the 10th International Symposium on Distributed Autonomous Robotic Systems, STAR, vol. 83, pp. 203–216 (2013)
12. Alonso-Mora, J., Breitenmoser, A., Beardsley, P., Siegwart, R.: Reciprocal collision avoidance for multiple car-like robots, In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 360–366 (2012)
13. Tan, J., Xi, N., Sheng, W., Xiao, J.: Modeling multiple robot systems for area coverage and cooperation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2568–2573 (2004)
14. Santos, V.G., Campos, M.F.M., Chaimowicz, L.: On segregative behaviors using flocking and velocity obstacles. In: Proceedings of the 11th International Symposium on Distributed Autonomous Robotic Systems, STAR, vol. 104, pp. 121–133 (2014)
15. He, L., van den Berg, J.: Meso-scale planning for multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2824–2829 (2013)
16. Derenick, J., Michael, N., Kumar, V.: Energy-aware coverage control with docking for robot teams. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3667–3672 (2011)

Distributed Safe Deployment of Networked Robots

Reza Javanmard Alitappeh and Luciano C.A. Pimenta

Abstract In real applications, it is always important to consider the generation of safe paths for robots during deployment or in future excursions through the environment. In order to include *safety* in the problem of deploying mobile robotic networks, we propose a new strategy based on the locational optimization framework. Our approach models the optimal deployment problem as a constrained optimization problem with inequality and equality constraints. This optimization model is built by incorporating into the locational optimization framework new features such as the classical Generalized Voronoi Diagram (GVD) commonly used as a safe roadmap in the context of path planning and a new metric to compute distance between robots and points in the environment. This new metric induces a new Voronoi partition of the environment. Furthermore, inspired by the classical Dijkstra algorithm, we present a novel efficient distributed algorithm to compute solutions in complicated environments.

Keywords Mobile robotic network · Locational optimization · Deployment problem · Voronoi partitioning

1 Introduction

According to [4], a system composed of a group of robots that sense their own position, exchange messages following a communication topology, process information, and control their motion is called a robotic network. One can find several applications for this type of system such as surveillance, sensing coverage, environment monitoring, search and rescue, etc. An important question to answer when using a robotic network is where each robot should be placed in the environment. In the

R.J. Alitappeh (✉) · L.C.A. Pimenta
Universidade Federal de Minas Gerais, Av. Antonio Carlos 6627, Belo Horizonte, MG 31270-901, Brazil
e-mail: rezajavanmard64@gmail.com

L.C.A. Pimenta
e-mail: lucpim@cpdee.ufmg.br

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_5

present work we show a distributed solution to this question which is refereed as the deployment problem [4]. The solution is distributed in the sense that each agent depends only on information from a small set of other agents called neighbors to compute its actions. Besides, this set of neighbors is dynamic since it might change as the system evolves. As pointed by [7], this allows for scalability and robustness. We are interested in finding optimal deployment configurations. We consider that a configuration is optimal if it is a minimizer of a functional encoding the quality of the deployment. This quality of deployment is related to the time of response of the network after an event that needs servicing happens in the environment. This time is a function of the distance of the agents from the event and the agent capabilities (speed, sensor field of view, etc.). In order to minimize the distance between agents and events, our approach applies the idea of partitioning the environment into subregions which are then assigned to specific agents. Therefore, each agent is responsible for attending the events in its corresponding subregion. Differently from previous works found in the literature we are also concerned with the incorporation of some safety constraints into the deployment. This property guarantees such a safe movement for the robots, with maximum distance from the obstacles, in the environment. The paper is organized as follows: in the next section we present some related work. In Sect. 3 we present some useful tools that will be considered in the rest of the paper. The proposed optimization model and an efficient distributed solution to the problem are explained in Sects. 4 and 5. Implementation results are shown in Sect. 6. Finally, conclusions are presented in Sect. 7.

2 Related Work

Our approach builds on the work in [7]. The authors of this work present a distributed and asynchronous approach for optimally deploying a uniform robotic network in a domain based on a framework for optimized quantization derived in [11]. Each agent (robot) follows a control law, which is a gradient descent algorithm that minimizes the functional encoding the quality of the deployment. Further, this control law depends only on the information of position of the robot and of its immediate neighbors. Neighbors are defined to be those robots that are located in neighboring Voronoi cells. Besides, these control laws are computed without the requirement of global synchronization. The functional also uses a *distribution density function* which weights points or areas in the environment that are more important than others. Thus it is possible to specify areas where a higher density of agents is required. This is important if events happen in the environment with different probabilities in different points. Furthermore, this technique is adaptive due to its ability to address changing environments, tasks, and network topology. Different extensions of the framework devised in [7] have been proposed in the literature. The problem of considering time-varying distribution density functions was studied in [13] to solve a task of simultaneous coverage and intruders tracking. Deployment and exploration in non-convex environments was considered in [3, 5, 10]. In [12], heterogeneous robots in

a non-convex environments were taken into account. Where, instead of point robots, disc shaped robots were considered. Some works also considered the discretization of the environment by grid cells to facilitate computation in complex environments. In [9] the authors consider a discrete partitioning and coverage optimization algorithm for robots with short-range communication. In this case a discrete setup was presented in which a discrete deployment functional is defined. The authors proved that their algorithm converges to a subset of the set of centroidal Voronoi tessellations (CVT) in discrete formulation, named pairwise-optimal partition. Gossip communication was used to allow information exchange among the agents. Similarly, [14] describe an algorithm to solve the deployment problem in a discrete setup. In [2] the environment was also discretized to allow the numerical computation of the environment partition (geodesic Voronoi diagram), but in this case the context was the one of generating an approximation to the continuous setup. In the same spirit of approximating the continuous setup, the authors of [1] discretized the environment and used a graph based approach inspired by Dijkstra algorithm [8] to directly compute the proposed control law in an efficient manner in general Riemannian manifolds with boundaries.

Statement of Contributions: The present paper further extends the works in [1, 12] to include safety. By merging different Voronoi diagrams, including the well known Generalized Voronoi Diagram (GVD) [6] (traditionally used as a roadmap in path planning) and by considering a constrained optimization problem in the context of the Locational Optimization Framework, we can generate safe routes for the robots during deployment and also after deployment when servicing a given point of the environment. We propose a new Voronoi Diagram which is built according to a new metric that takes into account shortest paths that traverse the GVD. Moreover, in order to consider real world environments we devise a new efficient algorithm to compute the next actions of the robots in the same spirit of the one in [1].

3 Background

In this section we explain the basic tools which will allow us to define our deployment problem in terms of a constrained optimization problem. These tools are the GVD and the locational optimization framework.

3.1 Generalized Voronoi Diagram

Let the set of obstacles $\mathbf{QO} = \{\mathcal{QO}_1, \ldots, \mathcal{QO}_n\}$ in a planar configuration space be called sites. This set induces a structure called Generalized Voronoi Diagram (GVD). $\mathcal{Q}$ indicates to configuration space and a set of points in the free configuration space, $\mathcal{Q}_{free}$, is defined as the Voronoi region of the obstacle $\mathcal{QO}_i$, V_i, if these points are

closer to $\mathcal{QO}_i$ than to all the other sites. Given an obstacle $\mathcal{QO}_i$, the generalized Voronoi region, V_i, is the closure of the set of points closest to $\mathcal{QO}_i$ [6].

$$V_i = \{\mathbf{q} \in \mathcal{Q}_{free} |\, d(\mathbf{q}, \mathcal{QO}_i) \leq d(\mathbf{q}, \mathcal{QO}_j), \forall j \neq i\}\,, \tag{1}$$

where $d(\mathbf{q}, \mathcal{QO}_i)$ shows the minimum distance between $\mathcal{QO}_i$ and $\mathbf{q}$. The two-equidistant surjective surface, $\mathcal{L}_{i,j}$ is the set of points equidistant to two obstacles $\mathcal{QO}_i$ and $\mathcal{QO}_j$ with distinct gradient vectors:

$$\mathcal{L}_{i,j} = \{\mathbf{q} \in \mathcal{Q} | d(\mathbf{q}, \mathcal{QO}_i) = d(\mathbf{q}, \mathcal{QO}_j) \text{ and } \nabla d(\mathbf{q}, \mathcal{QO}_i) \neq \nabla d(\mathbf{q}, \mathcal{QO}_j), j \neq i\}. \tag{2}$$

The points in $\mathcal{L}_{i,j}$ that are part of the GVD are those in which $\mathcal{QO}_i$ and $\mathcal{QO}_j$ are the closest obstacles. Therefore we can define the set:

$$V_{i,j} = \{\mathbf{q} \in \mathcal{L}_{i,j} |\, d(\mathbf{q}, \mathcal{QO}_i) \leq d(\mathbf{q}, \mathcal{QO}_h)\}. \tag{3}$$

This last definition allows us to formally define the GVD:

$$GVD = \bigcup_i \bigcup_j V_{i,j}. \tag{4}$$

An interesting feature of the GVD is that it can be used as a roadmap (RM) for path planning (Fig. 1a).

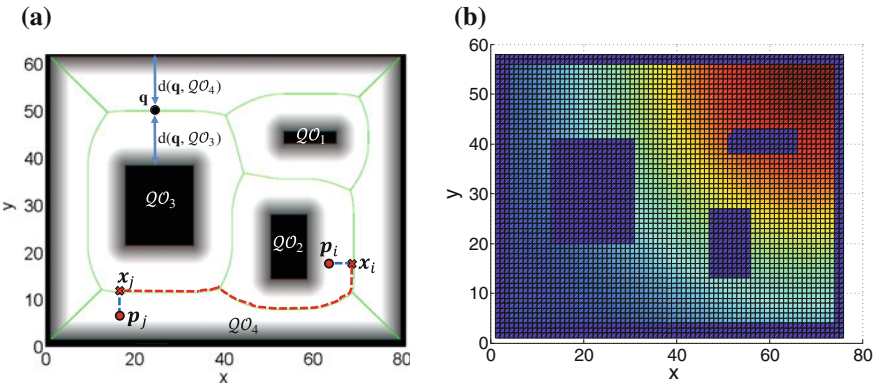


Fig. 1 A map with obstacles, $\mathcal{QO}_i$, GVD and a simple path on the GVD, to show its property to be used as a roadmap and a density function which centered at top-right of the map. **a** *Green line* shows the GVD. **q** is an equidistant point between sites $\mathcal{QO}_3$ and $\mathcal{QO}_4$. *Dash line* illustrates a path between two arbitrary points, (p_i, p_j). **b** An example of Gaussian density function in a 2D map. $A = 7,\ (x_0, y_0) = (67, 54),\ \sigma_x = \sigma_y = \sqrt{30}$

3.2 Locational Optimization Based Deployment

Let $\Omega \subset \mathbb{R}^2$ be the map of the environment. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the configuration of n robots, and $f(d(\mathbf{q}, \mathbf{p_i}))$ indicates the cost of servicing an event at point $\mathbf{q}$ by robot i. This is related to the spatial distance between $\mathbf{q}$ and $\mathbf{p}_i$ as $d(\mathbf{q}, \mathbf{p}_i)$ represents this distance and f is a smooth strictly increasing function. Suppose we have access to a density function $\phi : \Omega \to \mathbb{R}_+$ which gives weights to different points in Ω to reflect the probability of having events at each point (See Fig. 1b). Considering also the tessellation $W = \{W_1, \ldots, W_n\}$ so that $\cup_{i=1}^{n} W_i = \Omega$ and $I(W_i) \cap I(W_j) = \emptyset$, $\forall i \neq j$, where $I(\cdot)$ represents the interior of a given region, it is possible to define the following *deployment functional* that measures the quality of the robotic deployment [7]:

$$\mathcal{H}(P, W) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i, W_i) = \sum_{i=1}^{n} \int_{W_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q}\,, \tag{5}$$

The objective of the Locational Optimization based framework is to drive the robots to a configuration that minimizes (5). In [1], it is considered the case where f is the square function and d is the Euclidean distance. The authors of [1] proposed a distributed control law that guides these robots to the minimum which coincides with the so-called Centroidal Voronoi Tessellation (CVT).

In the present work, we further extend this framework to incorporate safety.

4 Safe Deployment Modeling

In this section we define the safe deployment problem as an optimization problem in the context of the Locational Optimization Framework. Consider the bounded free configuration space $\mathcal{Q}_{free} \subset \mathbb{R}^2$. Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the configuration of n robots, where $\mathbf{p}_i \subset \mathcal{Q}_{free}$. The problem to be solved is the one of finding distributed robotic actions, in the sense that only robots in the neighborhood of robot i will be taken into account, that leads the system to a local solution of the constraint minimization problem given below:

$$\min_{\mathbf{p}_i} \mathcal{H}(P, GGVD), \qquad \text{s.t.} \begin{cases} y_{i1}(\mathbf{p}_i) \leq 0\,, \ldots\,, y_{im}(\mathbf{p}_i) \leq 0 \\ h(\mathbf{p}_i) = 0 \end{cases} \tag{6}$$

where $y()$ and $h()$ declare inequality and equality constraints respectively. The next sections will explain the meaning of the terms used in the defined problem and from this explanation it should be clear how safety is then incorporated in the locational optimization framework.

New Metric: The Geodesic distance is a metric which is more realistic than Euclidean distance in non-convex environments. This distance is used in the deployment functional presented in [12] as the general function d, as defined in the last section. In this case, the induced Voronoi Tessellation is the so-called geodesic Voronoi Tessellation. Now, we propose a further extension on this metric which will be called *Geodesic Distance Based* on GVD. This distance function corresponds to the length of the shortest path from two points when using a GVD as a roadmap. A clear example of this path is shown in Fig. 1a, where dash line between a pair (p_i, p_j) defines the whole path: $\{(p_i, x_i), (x_i, x_j), (x_j, p_j)\}$. In general, we can divide this path into three parts: a path from the initial point to GVD ($Path_{\ Init\_To\_GVD}$), a path from a point on GVD to another point on GVD ($Path_{\ GVD\_To\ GVD}$), and a path from GVD to the goal point ($Path_{\ GVD\_To\_Goal}$). The *Geodesic Distance Based on GVD* is then defined as:

$$\begin{aligned} d(\mathbf{p}_i, \mathbf{p}_j) \;=\; & W_1.||\mathbf{p}_i - \Pi_i(GVD)|| + W_2.g(\Pi_i(GVD), \Pi_j(GVD)) \\ & + W_1.||\mathbf{p}_j - \Pi_j(GVD)||\,, \end{aligned} \tag{7}$$

where $g(x_i, x_j)$ gives the shortest distance between two points x_i and x_j on the GVD, if the motion is constrained to remain on the GVD, $\Pi_i(GVD)$ represents the projection of the point $\mathbf{p}_i$ onto the GVD which corresponds to the closest point on the GVD to $\mathbf{p}_i$, and W_1 and W_2 are the weights of each part of the path. For example, by assigning a big value to W_1 the cost of $Path_{\ Init\_To\_GVD}$ or $Path_{\ GVD\_To\_Goal}$ can be increased. These weights help to adjust the cost of two portions of the path so that it is worth first moving to the GVD as soon as possible and perform most of the motion traversing it. As safety regarding the existing obstacles is related to the distance the robot keeps from them and the GVD provides a roadmap which keeps equidistance from the closest obstacles, we can say that this metric can introduce safety in the deployment solution. In the minimization problem defined in (6) the cost function is defined according to the new metric, d:

$$\mathcal{H}(P, GGVD) = \sum_{i=1}^{n} \int_{GGVD_i} d(\mathbf{q}, \mathbf{p}_i)^2 \phi(\mathbf{q}) d\mathbf{q}\,, \tag{8}$$

where *GGVD*, (Geodesic Generalized Voronoi Diagram) will be defined as the Voronoi Tessellation induced by the new metric and $W_1 >> W_2$.

Now, we can also describe the equality constraint $h(\mathbf{p}_i) = 0$. This function is defined as the difference between the distance functions $d(\mathbf{p}_i, \mathcal{QO}_i)$ and $d(\mathbf{p}_i, \mathcal{QO}_j)$ in which $\mathcal{QO}_i$ and $\mathcal{QO}_j$ are the closest obstacles to robot i. Thus, this means the robots must be deployed along the GVD.

Collision avoidance: Since the focus of this work is on safety, besides the static obstacles we should also take into account the possible collisions between robots. A practical problem of the unconstrained minimization executed by the pure gradient-descent law in [7] is that actual robots are not point-robots. Thus, we propose to use here the same strategy presented in [12]. Basically, in this work the basic results for point robots are extended to robots that can be modeled as circular disks, each one

with radius $r_{\mathbf{p}_i} = r_{\mathbf{p}_j}, \forall i, j$. This is done by incorporating the inequality constraints $y_{ik} \leq 0$ in (6), so that the robots remain inside their so-called free Voronoi region. For details refer to [12].

5 Proposed Solution

In order to solve the safe deployment problem in an efficient manner we propose to solve a discrete approximation of the continuous setup shown in the last section. The proposed algorithm builds upon the work in [1] which presents a modified Dijkstra algorithm able to compute simultaneously at each iteration the geodesic Voronoi diagram and the robot next actions in the case of deployment on Riemannian manifolds with boundaries.

5.1 Discrete Approximation

Consider the 2-dimensional configuration space. The graph $G = \{\mathcal{V}(G), \mathcal{E}(G), \mathcal{C}_G\}$ is induced from the uniform square tiling of the configuration space by considering an 8-connectivity neighborhood (See Fig. 2a). The set of vertices (nodes) is given by $\mathcal{V}(G)$, the set of edges by $\mathcal{E}(G) \subseteq \mathcal{V}(G) \times \mathcal{V}(G)$, and a cost function is denoted by $\mathcal{C}_G : \mathcal{E}(G) \rightarrow \mathbb{R}^+$. It is important to mention that a node of the graph is placed in grid cells located inside $\mathcal{Q}_{free}$. Moreover, the cost of each edge is computed based on the defined new metric as will be clarified later. We will also use the notation p_i to denote the node that contains the position of robot i, $\mathbf{p}_i$, and the operator $\mathbf{P}(s)$ to

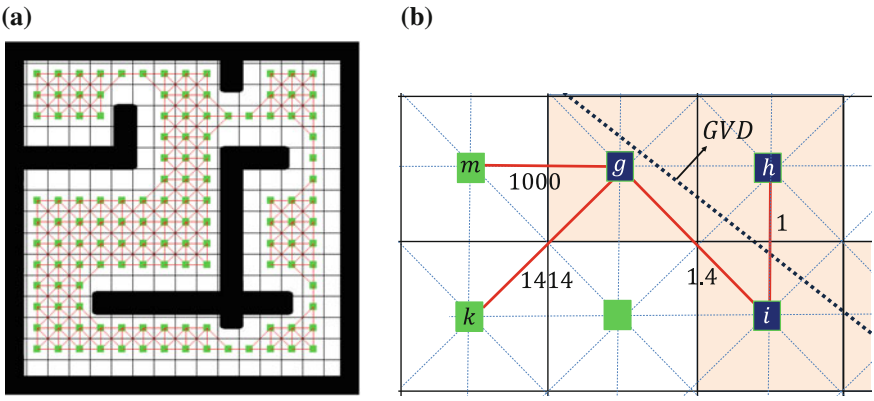


Fig. 2 Representing a discrete grid based map and the technique of modeling GVD based on graph nodes and edges. **a** Discretization and graph representation. **b** The nodes g, h, and i are in the set $\mathcal{V}GVD(G)$

return the position of the center of the grid cells. Therefore, $\mathbf{P}(p_i)$ returns the center of the cell that contains robot i. Furthermore, we will use $\mathcal{N}_G(p_i)$ as the set of graph vertex neighbors: $\mathcal{N}_G(p_i) = \{q \in \mathcal{V}(G) \mid [p_i, q] \in \mathcal{E}(G)\}$.

We compute the GVD before discretizing the environment into cells, allowing the GVD to be independent from the discretization resolution. The GVD is embedded in our graph by labeling the set of grid cells that contain a piece of the GVD as the approximate GVD, $\mathcal{V}GVD(G)$. Now, we can define the edge cost function:

$$\mathcal{C}_G(i,j) = \begin{cases} W_2 \cdot c(i,j), & \text{if } i\,, j\ \in \mathcal{V}GVD(G) \\ W_1 \cdot c(i,j), & \text{otherwise}\,, \end{cases} \tag{9}$$

where $c(i, j)$ is given by the Euclidean distance between the centers of the cells i and j. Since it is our objective to deploy and also move the robots along the GVD we will use $W1 >> W2$. For instance, see an example in Fig. 2b.

The shortest path between two vertices s and q corresponds to the sequence of nodes (consequently edges), $\{s, v_1, v_2, \ldots, v_m, q\}$, connecting this pair such that the sum of the edge costs is minimum. We will define this minimum cost sum as $d^*(\mathbf{P}(s), \mathbf{P}(q))$:

$$d^*(\mathbf{P}(s), \mathbf{P}(q)) = \mathcal{C}_G(s, v_1) + \mathcal{C}_G(v_1, v_2) + \ldots + \mathcal{C}_G(v_m, q). \tag{10}$$

This allows us to define the discrete version of the deployment functional:

$$\mathcal{H}^* = \sum_{i=1}^{n} \sum_{q \in GGVD_i^*} d^*(\mathbf{P}(q), \mathbf{P}(p_i))^2 \phi(\mathbf{P}(q)) \bar{w}\,, \tag{11}$$

where $GGVD_i^*$ corresponds to the set of grid cells so that $d^*(\mathbf{P}(q), \mathbf{P}(p_i))$ is less than $d^*(\mathbf{P}(q), \mathbf{P}(p_j))$, $\forall j \neq i$, and $\bar{w}$ is a constant related to the integral element of area. Assuming that the robots are located at the center of the grid cells, i.e. $\mathbf{P}(p_i) = \mathbf{p}_i$, we can compute the gradient of $\mathcal{H}^*$:

$$\frac{\partial \mathcal{H}^*}{\partial \mathbf{p}_i} = \sum_{i=1}^{n} \sum_{q \in GGVD_i^*} 2\mathbf{z}_{p_i,q} d^*(\mathbf{P}(q), \mathbf{P}(p_i)) \phi(\mathbf{P}(q)) \bar{w}\,, \tag{12}$$

where $\mathbf{z}_{p_i,q}$ is the vector with direction given by the first edge of the shortest path between p_i and q, i.e. the direction of $\mathbf{P}(p_i) - \mathbf{P}(v_1)$, and magnitude given by W_2 if $p_i\,, v_1\ \in \mathcal{V}GVD(G)$ or W_1 otherwise. Based on last equation we propose a gradient descent based approach in the next subsection.

5.2 Distributed Algorithm

The problem defined in (6) has some constraints, which means that our solution should also take these constraints into account to define the next action. The collision avoidance inequalities are implemented by first verifying if any of these constraints are active, i.e., $y_{ik} = 0$ for some k. If this is the case, it means there are at least two robots in the imminence of a collision, thus the involved robots will be allowed to move only if the desired direction of motion is orthogonal or has a negative projection onto the segment joining the two robot centers. The equality constraint which enforces the robots to be deployed along the GVD is imposed in our solution by means of two steps. If a robot is not in a cell that is part of the $\mathcal{V}GVD$ the next action for this robot is to move towards the closest cell in $\mathcal{V}GVD$ which is not occupied by any other robot at that time. This can be considered as the first step of the proposed approach. The second step is activated when the robot enters a cell which is part of the $\mathcal{V}GVD$. Now, the next action of this robot is a motion along a straight line from the current grid cell to a neighbor cell which is also part of the $\mathcal{V}GVD$. This next cell is computed based on the gradient descent direction given by the negative of the expression in (12). As in [1] we present an algorithm that computes the gradient descent direction and the Voronoi tessellation, $GGVD^*$, simultaneously in every time-step by means of a wavefront propagation procedure similarly to the process in Dijkstra's algorithm [8]. The wavefront in a given iteration represents the set of points equidistant to the start node also called source. In our case we consider wavefronts emanating from multiple sources (given by the locations of the robots). As the wavefronts propagate two operations are executed: (i) graph vertices in the wavefronts are associated to robots (sources) at shortest distance (according to the proposed metric) giving rise to the Voronoi regions; and (ii) terms of the summation in (12) associated to vertices in the wavefronts are added to a variable responsible to store the gradient descent direction. The places where the wavefronts collide determine the Voronoi boundaries. The ideas previously discussed are organized in the form of the Algorithms 1 and 2. We consider these are the algorithms running in robot i.

Termination: The commands in while loop in Algorithm 1 are executed until termination criteria are met. An interesting criteria is the observation of the variation of positions of robots in the most recent iterations. If this variation is below a given threshold the algorithm can terminate.

Complexity: It is clear that the bottleneck of our iterative algorithm is the function described in Algorithm 2. Since this function runs exactly in the same format of the Dijkstra algorithm, the graph vertices have a constant degree, and a heap is maintained as a priority queue to store the unvisited nodes, the running time is given by $O(V_G \log(V_G))$ (where V_G is the number of vertices in the graph).

Algorithm 1: Distributed main algorithm running in robot i.

Input: $G, \mathcal{VGVD}, \phi, p_i$
G is the graph, $\mathcal{VGVD}$ is the approximate generalized Voronoi diagram, ϕ is the density function, $p_i \in \mathcal{V}_G$ is robot i initial location (graph node).
Output: p_i, o
p_i is robot i final location and $o : \mathcal{V}_G \rightarrow \{1, 2, \ldots, n\}$ is the discrete tessellation map $GGVD^*$ as computed by robot i.

while *(Termination criteria is not met)* **do**
  Broadcast position $\mathbf{p}_i$ //Robot i sends its position to other robots.
  $\mathcal{N}_i \leftarrow$ *Get_Location_of_Neighbor_Robots*() //Robot i receives location of other robots.
  $\mathcal{N}_i^* \leftarrow \mathcal{N}_i \cap \mathcal{VGVD}$ // Set of neighbors already in the $\mathcal{VGVD}$.
  if $p_i \notin \mathcal{VGVD}$ **then** //check if the robot is not on the $\mathcal{VGVD}$.
    Set the current direction of motion as the one towards the closest cell in $\mathcal{VGVD}$ which is not occupied by another robot
  else
    Call *Modified_Dijkstra*$(G, \mathcal{VGVD}, \phi, p_i, \mathcal{N}_i^*)$ //Compute both the next action (cell) p_i' and the $GGVD^*$ as seen by robot i.
    Set the current direction of motion to reach p_i'
  if *(There is no active inequality constraint) OR (There is an active inequality constraint AND current direction of motion is not obstructed by another robot* **then** //collision avoidance constraint.
    Move according to the current direction of motion
  else
    Stop

6 Results

In this section we illustrate our approach by simulating the deployment of robots in two different environments. Videos are available at:
http://www.cpdee.ufmg.br/~coro/movies/DARS2014/

Simple map: A simple room with some obstacles and size 3.79×2.91 m (or image with 728×582 pixels) is the input map. By using a discretization resolution equal to 10 pixels, we have a grid map with 72×58 cells. Discretization rate can be defined based on the real size of robot and map. Density function is defined by a Gaussian function with parameters: $(x_0, y_0) = (67, 54)$, $\sigma_x = \sigma_y = \sqrt{30}$. In this experiment 5 robots are considered. By observing robots' movement during deployment, it is evident at the beginning, two robots have a large Voronoi region when other robots do not have it (See Fig. 3b, c). After some iterations the decrease/increase of size of large/small Voronoi regions contribute to minimize the cost function as it is shown in Fig. 3a.

Office-like map: In the second experiment, the method was tested on a more complicated map with size 40.0×60.0 m and grid graph size of 80×120. Initially, some of the robots are on the GVD and others are not. We define density Gaussian function as: $(x_0, y_0) = (10, 110)$, $\sigma_x = \sigma_y = \sqrt{50}$. Because of the large input map,

Algorithm 2: *Modified_Dijkstra()function.*

Input: $G, \mathcal{VGVD}, \phi, p_i, \mathcal{N}_i^*$
G is the graph, $\mathcal{VGVD}$ is the approximate generalized Voronoi diagram, ϕ is the density function, $p_i \in \mathcal{V}_G$ is the current robot i location (graph node), and $\mathcal{N}_i^*$ is the set of locations of other robots already in $\mathcal{VGVD}$.
Output: p_i', o
p_i' is the next cell for robot i and $o : \mathcal{V}_G \rightarrow \{1, 2, \ldots, n\}$ is the discrete tessellation map $GGVD^*$ as computed by robot i.

Initiate d^*: $d^*(v) \leftarrow \infty$, for all $v \in \mathcal{V}(G)$ // New metric distance.
Initiate o: $o(v) \leftarrow -1$, $\forall v \in \mathcal{V}(G)$ // Tessellation.
Initiate η: $\eta(v) \leftarrow \emptyset$, $\forall v \in \mathcal{V}(G)$ // robot graph vertex neighbor. $\eta : \mathcal{V}(G) \rightarrow \mathcal{V}(G)$
$\mathbf{I}_i \leftarrow \mathbf{0}$ // The gradient descent of the discrete functional.
foreach $i \in p_i \cup \mathcal{N}_i^*$ **do**
    $d^*(p_i) \leftarrow 0$
    $o(p_i) \leftarrow i$
    foreach $q \in \mathcal{N}_G(p_i)$ **do** // For each graph vertex neighbor of p_i
        $\eta(q) \leftarrow q$
$Q \leftarrow \mathcal{V}(G)$ // Set of unvisited nodes.
while $(Q \neq \emptyset)$ **do**
    $q \leftarrow \arg\min_{q' \in Q} d^*(q')$ // Maintained by a heap data-structure.
    $Q \leftarrow Q - q$ // Remove q from Q
    $k \leftarrow o(q)$
    $s \leftarrow \eta(q)$
    if $(s \,!= \emptyset)$ *AND* $(k == p_i)$ **then** // Equivalently, q is not a vertex occupied by a robot and $q \in GGVD_i^*$.
        $\mathbf{I}_i \leftarrow \mathbf{I}_i + \phi(q) \times d^*(q) \times (\mathbf{P}(s) - \mathbf{P}(p_i))$
    foreach $w \in \mathcal{N}_G(q)$ **do** // For each graph vertex neighbor of q
        $d' \leftarrow d^*(q) + \mathcal{C}_G(q, w)$ //relaxation.
        if $d' < d^*(w)$ **then**
            $d^*(w) \leftarrow d'$
            $o(w) \leftarrow k$
            if $(s \,!= \emptyset)$ **then**
                $\eta(w) = s$
$p_i' \leftarrow \arg\max_{u \in \mathcal{N}_G(p_i) \cap \mathcal{VGVD}} \frac{(\mathbf{P}(u) - \mathbf{P}(p_i))}{\|\mathbf{P}(u) - \mathbf{P}(p_i)\|} \cdot \mathbf{I}_i$ // Choose next action as the cell in $\mathcal{VGVD}$ best aligned with the gradient descent direction.

we consider three groups with two robots in each one. They start their movement from three different parts of the map. Figure 4a shows the final positions. Figure 4b illustrates robot trajectories and the evolution of the deployment functional, which is minimized as desired is depicted in Fig. 4c.

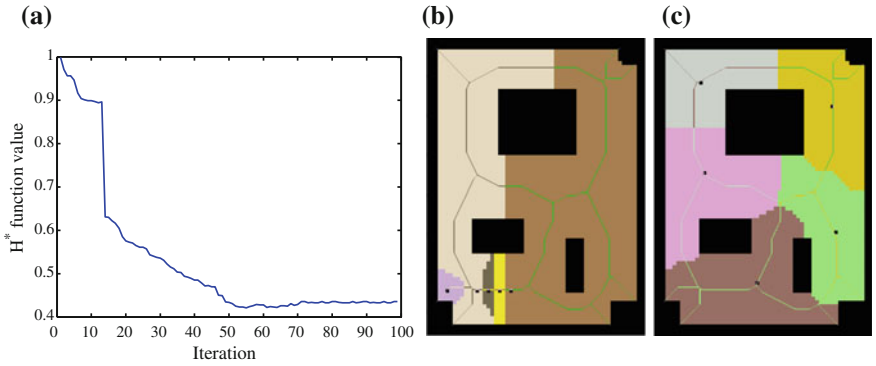


Fig. 3 Snapshots when running the proposed algorithm for 5 robots. **a** $\mathcal{H}^*$ function converged after 65 iterations. **b** *iter*1. **c** *iter*65

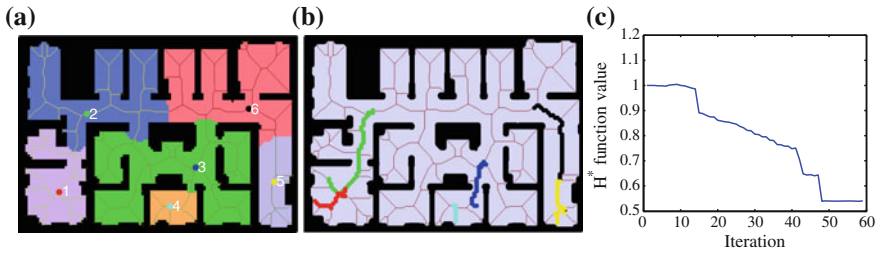


Fig. 4 Result of running the algorithm on office-like map. **a** *iter*50. **b** Robot trajectories in the office-like environment. **c** $\mathcal{H}^*$ function converged after 50 iterations

7 Conclusion

We addressed the problem of deriving optimal distributed control laws to deploy robotic networks in complex environments safely. The deployment problem is translated to a constrained optimization problem so that a deployment functional defined with the use of a new distance function must be minimized while satisfying constraints of two types: (i) inequality constraints for inter-robot collision avoidance, and (ii) an equality constraint to enforce the robots to be deployed at the generalized Voronoi diagram of the environment for maximizing distance from static obstacles. It is also interesting to mention that the proposed framework can also be used with other roadmaps different from the GVD. We presented a distributed algorithm strongly based on the one proposed in [1] which allows for efficient computation of a discrete solution for the discrete approximation of the problem. Simulations were also presented to illustrate the proposed method performance.

Acknowledgments We gratefully acknowledge support from CNPq and FAPEMIG.

References

1. Bhattacharya, S., Ghrist, R., Kumar, V.: Multi-robot coverage and exploration on Riemannian manifolds with boundaries. Int. J. Robot. Res. **33**(1), 113–137 (2013)
2. Bhattacharya, S., Michael, N., Kumar, V.: Distributed coverage and exploration in unknown nonconvex environments. In: Proceedings of the 10th International Symposium on Distributed Autonomous Robotics Systems, pp. 1–14. Springer (2010)
3. Breitenmoser, A.: Voronoi coverage of non-convex environments with a group of networked robots. In: The IEEE International Conference on Robotics and Automation (ICRA), pp. 4982–4989. IEEE, Anchorage, Alaska (2010)
4. Bullo, F., Cortés, J., Martinez, S.: Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms. Applied Mathematics Series. Princeton University Press, Princeton (2009)
5. Caicedo-Nunez, C.H., Zefran, M.: A coverage algorithm for a class of non-convex regions. In: Proceeding of the IEEE Conference on Decision and Control, pp. 4244–4249 (2008)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, Boston (2005)
7. Cortes, J., Martinez, S., T.K., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. **20**(2), 243–255 (2004)
8. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische mathematik **1**(l 959), 269–271 (1959)
9. Durham, J.W., Carli, R.: Discrete partitioning and coverage control for gossiping robots. IEEE Trans. Robot. **28**(2), 364–378 (2012)
10. Haumann, D., Breitenmoser, A., Willert, V., Listmann, K., Siegwart, R.: DisCoverage for non-convex environments with arbitrary obstacles. In: Proceeding of IEEE International Conference Robotics Automation, pp. 4486–4491 (2011)
11. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
12. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In: IEEE Conference on Decision and Control 2, pp. 3947–3952. IEEE, Cancun, Mexico (2008)
13. Pimenta, L.C.A., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R.C., Pereira, G.A.S.: Simultaneous coverage and tracking (SCAT) of moving targets with robot networks. In: Algorithmic Foundation of Robotics VIII, Springer Tracts in Advanced Robotics **57**, 85–99 (2010)
14. Yun, S.k., Rus, D.: Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. Robotica **32**(02), 257–277 (2013)

MarSim, a Simulation of the MarsuBots Fleet Using NetLogo

David Leal Martínez and Aarne Halme

Abstract The Marsubots fleet is an heterogeneous robot fleet consisting of Marsus and Motherbots, the purpose of this fleet is to explore previously unexplored areas as well as partially explored areas or areas that have suffered alterations. In order to be able to explore large areas such as large buildings and open spaces, the robots need to recharge their batteries from the Motherbot's recharging bay. This paper focuses on describing the simulation environment MarSim that has been created using NetLogo to model the fleet in order to be able to use tools like Genetic Algorithms to refine the parameters that have been identified as key parameters for the robots to complete the task at hand successfully, specially after a large amount of recharge cycles.

Keywords Distributed robotics · MarsuBots · NetLogo · Power management · Recharging

1 Introduction

Creating a map with multiple robots has been in roboticists minds for a while and already suitable algorithms to create and merge the maps created by different robots have been created. Now in order to take this knowledge into the field, some constrains have to be taken into account, such as power management. While mapping, robots will eventually run out of power and they will have to either stop mapping to go and re-charge themselves or need a re-charging unit go and re-charge them before or after they run out of power.

D. Leal Martínez (✉) · A. Halme
Department of Automation and Systems Technology, Aalto University School of Electrical Engineering, Otaniementie 17, Espoo, Finland
e-mail: david.leal@aalto.fi

A. Halme
e-mail: Aarne.Halme@aalto.fi

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_6

1.1 The Simulator and the Fleet Behind It

In this paper an approach to this recharging problem will be shown with a model done in NetLogo multi agent simulator. The simulation is based on the MarsuBots robot society built at the Automation and Systems Technology Laboratory of Aalto University. It consists of a Mother Robot, which is a Tank-like robot that can host up to 3 smaller Marsu robots inside, and a fleet of 6–8 Marsu robots. The Mother Robot or Motherbot (MB) has the capability of carrying Marsus inside of it, charging their batteries, climbing stairs with robots inside as done in [4] and has higher computing power and sensor capabilities than the smaller Marsus as described in [5]. The Marsu robots are smaller two wheeled robots that have recently been upgraded to run on a Beaglebone board capable of serial communications, Ethernet and Wi-Fi for network/internet access, i2c bus and can run at 1 GHz (2000 MIPS). Marsus have a sensor set including a Laser range finder, a camera, 4 ultrasonic sensors and encoders on the wheels for odometry, but they also have the ability to re-charge another unit via the recharging port. In Fig. 1 a Marsu unit can be seen exiting the Motherbot via the front ramp. The fleet of Marsus has been the testbed of bearing-only SLAM methods for simultaneous localization and mapping as reported in [3].

A fleet of robots is usually limited by size of their batteries and their ability to recharge them, which is normally done in recharging stations or battery swapping techniques. The MarsuBot fleet has been designed to minimize this limitation by having the Motherbot behave as a moving recharging station, this way allowing the fleet to explore spaces as long as the reserves of the Motherbot are not depleted.

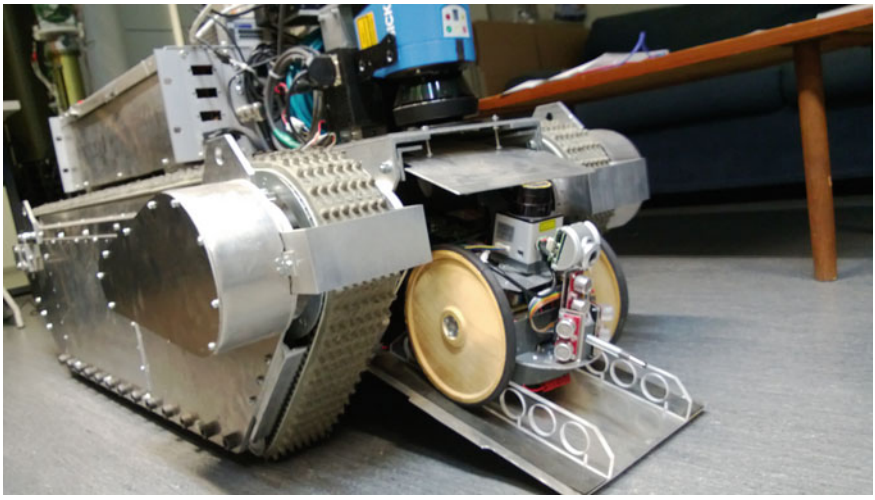


Fig. 1 Marsu unit leaving the MB via the front ramp

In Chap. 2 we will discuss the simulation model, in Chap. 3 the results acquired and in the 4th chapter the future implementation of this algorithm in the real fleet.

2 Simulation

In this chapter the simulation model implementation as well as the algorithms used will be presented starting with a basic introduction to the NetLogo multi-agent simulator and the general scenario being simulated, followed by the description of the state machine and its corresponding states.

2.1 NetLogo

Net logo is a multi-agent programmable modeling environment popular for both teaching and implementing real life multi-agent scenarios. It has been authored by Uri Wilensky and developed at the CCL and its available free of charge. For more information refer to [7]. In the NetLogo environment, the agents are divided into turtles, patches, links and the observer. In this model only turtles (robots being either Marsus or MB) and patches (forming the grid representing the world in which the turtles live and act) will be used. In the simulation every Marsu is shaped like an arrow, and has its own unique color assigned randomly every time a simulation is ran and the MB is represented by a gray colored tank that will show colored bars inside of it representing how many Marsus are inside. In Fig. 2 a sample run can be seen depicting how the Marsus look, as well as the MB and what the patches color mean.

2.2 Scenario

In this model the goal for the robots is to be able to map the whole office environment without running out of energy while avoiding going through areas that other robots have already discovered and more importantly avoiding targeting the same temporal goal as other Marsus. As they travel throughout the world, the robots will be scanning their surroundings up to 5 patches away from themselves (this way simulating using a laser rangefinder with limiting ranging capability) and marking those as explored space by turning the color of the cells from black (empty space) to either green (explored space) or yellow (explored wall). The Exploration method used by the robots is to first `define` all the existing frontier cells (a frontier cell is a cell that is in the border between explored and non explored space as defined and used by [2]) then `choose` one of those cells as its target, `plan a path` to that cell avoiding walls, `follow` that path to the target so when reached a new target will be chosen. This cycle will continue until either the whole space is discovered or the Marsu's battery reserve

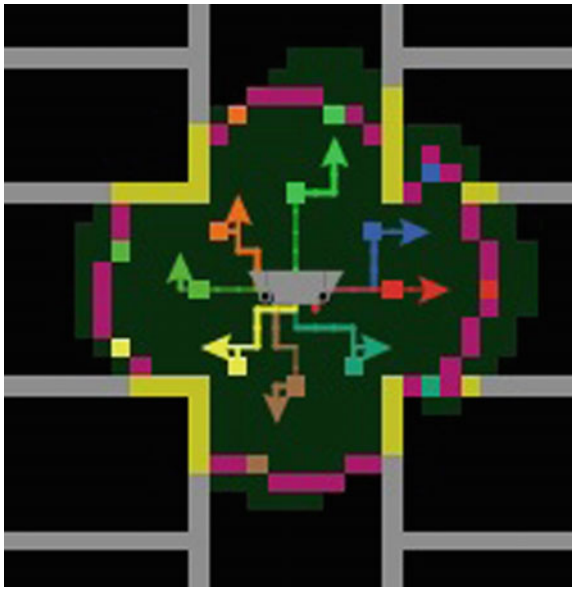


Fig. 2 20 Marsus starting to explore an unknown environment. The MB is shown in the center of the picture as the *gray* tank, while the Marsus are the arrow shaped agents with different colors. The *black* patches represent unexplored space while the *gray* ones unexplored walls, *green* patches are already explored space, *yellow* ones already explored walls, *magenta* patches are frontier patches and the multi colored patches are the goal patches of Marsus with matching colors

Table 1 Multi robot exploration algorithm

State	Action
1	Define frontier cells
2	Choose frontier cell
3	Plan path
4	Follow path
5	Go back to MB
6	Queue
7	Recharge

falls bellow the defined threshold in which case the robot will then `go back` to the MB, `queue` if needed, `recharge` batteries and then continue using the same exploration strategy. The state machine depicting the flow of the Marsu's behavior is presented in Table 1.

The Motherbot (MB) is considered to have a unlimited energy storage and can host up to three Marsus inside recharging at any given time.

2.2.1 Define Frontier Cell

A frontier cell is defined to be a cell that has at least one neighboring cell being undiscovered space, in the simulation this is achieved by asking all patches with undiscovered neighbors (black colored patches) to mark themselves as frontier cells and change their color to magenta as can be seen in Fig. 2.

2.2.2 Choose Frontier Cell

To choose a frontier cell, a Marsu looks to the cell that will provide the most utility for it. The total utility Ut is calculated by subtracting the cost of reaching the frontier cell Dc from the utility gained from reaching that goal cell Uc.

$$Ut = Uc - Dc; \tag{1}$$

Dc is measured in amount of cells between the goal and the current standing point, and both Ut and Uc are variables that are being studied in this research and will be commented in the results section.

After calculating the Ut for all frontier cells, the Marsu will then choose a target cell with the highest value by marking that square with its own color.

Once a Marsu has chosen a cell, its will alter the utility of the frontier cells surrounding it in the scan radius R, this way preventing other Marsus from choosing cells too close and spreading more evenly the exploration task as can be seen on Fig. 3 a similar approach as the one used in [2].

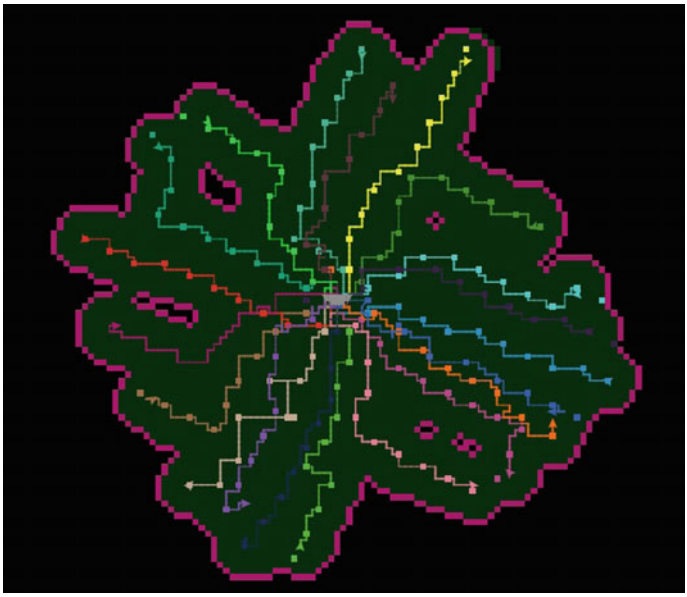


Fig. 3 Marsus moving in an unconstrained environment to demonstrate the algorithm used to choose frontier cells and spread evenly in the exploration effort

2.2.3 Plan Path

In order for the robot to travel from the actual cell to the already chosen frontier cell, a path must be planned. The strategy used to plan the path is to choose the lowest cost path, in order to find it, we first need to note that in this particular simulation the cost to travel from any cell to another is always 1 as long as neither of them is a wall. As the map is growing all the time by all the robots that are going into unexplored space it would be unwise to have static information about the cost to go from one cell to another, so the approach taken is that every time a robot needs to plan a path, it will calculate the cost by `flooding` the cost from the goal all the way into the actual position of the robot by making use of Bellman's principle of optimality [1]. `Flooding` consists of assigning a value of cost zero to the goal, and then that cell will ask all of its neighboring cells to set their cost to the cost of itself increased by 1, and ask those cells to ask their neighbors to repeat the process by asking their neighbors that are not walls and that dont have a value lesser than themselves (they would be the neighbors who were also neighbors to the goal in this first case) to do the same and do this process until the cell where the robot is standing get a value assigned, at this point the algorithm stops and the robot has a single path to follow by simply going to the neighbor with the smallest cost every single step. In many situations the robot will have more than one neighboring cell with the same value that is the smallest, in this particular case the robot chooses randomly one of them. In order to make it realistic and avoid that the robots drive next to the walls and have problems with the sharp corners a potential field is used, similar to the one discussed in Sect. 25.4 of [6] just that in this case, all the known wall cells ask their neighboring cells to make their cost infinite (or a number higher than the highest possible cost in the whole map space) before the flooding starts this way having a safe area around the known walls as can be seen in Fig. 4.

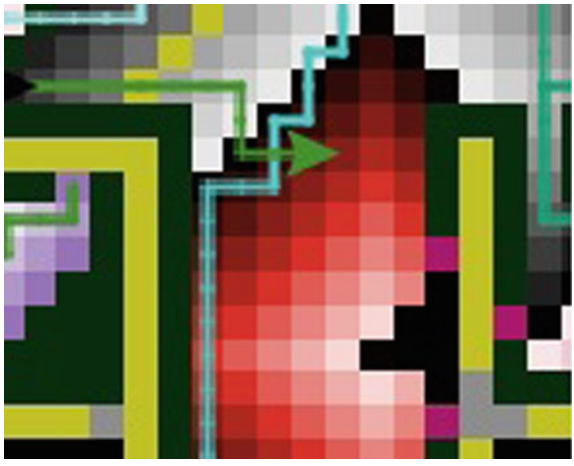


Fig. 4 Marsu using splash algorithm for reaching it's goal, in this figure yellow patches are already discovered walls, *gray* ones undiscovered walls, *green* ones discovered space evaded by algorithm and the gradient colored ones are the ones showing how the algorithm works by colorizing the value of the patch splash value, these are colored for illustration purposes only

2.2.4 Navigate Path

Once the path is planned the Marsu needs only to always choose the neighbors with the smallest cost and that will make it right to the goal cell. This would be that simple if there were no other Marsus driving around the area that they could collide into. To avoid collisions the Marsus always scan the front cell for a Marsu and in case one is there it will drive to the next cell on the right and then continue on its way, very similar to what you would do while driving a car, just drive on your right lane until the other car has passed and then continue on your way.

2.2.5 Go Back to Mother Robot

The Marsus are monitoring at all times their energy reserves and when the remaining amount of energy falls bellow a certain threshold (at this time set in the simulator's user interface) then the Marsu will start its journey to the location of the MB using the same `flooding` algorithm.

2.2.6 Approach and Queue

When a Marsu reaches the vicinity of the MB it will start its approach to dock with it, Queue if necessary and wait for its turn to recharge. When only 5 cells are left to reach MB will start the approach, meaning it will check if a queue exists, if its doesn't it will drive right into the MB and recharge, but if it detects a queue, it will request a queue number that will be assigned to it and then increased, and then go and wait 2 * the queue number cells away for the MB on it's east side as can be seen on Fig. 5. There it will wait until one unit exists the MB and when this happens, it will decrease its queue number by one and drive again to 2 * queue number cells away from the MB, and when its queue number is zero it will drive right into the MB.

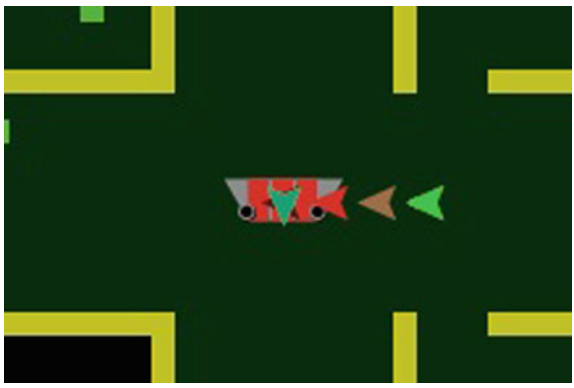


Fig. 5 Marsus queueing east of the MB while other Marsus recharge

2.2.7 Recharge

Once the Marsu is inside the MB it will recharge its battery until it is fully recharged and the exit the MB by choosing a new frontier cell and heading towards it by re entering state 1.

3 Results

The creation of this simulation proved NetLogo to be a suitable environment for testing the exploration algorithms in a simple yet realistic way that can be then be used to optimize certain parameters with the help of, for example, genetic algorithms that would would not be possible to test with the real robot fleet due to the fact that emptying a the battery of a working robot thousands of times could take too much time and resources to achieve.

By simulating the exploration algorithm to be used with the MarsuBot Fleet, key parameters were identified to be optimized using genetic algorithms so that the fleet can explore very large spaces that require a very large number of energy cycles. These parameters are:

Batt Threshold	The battery threshold that marks when is the best time for a Marsu to start heading back to the MB. Fine tuning this parameter using genetic algorithms it is expected to optimize the time a robot can be exploring new areas without risking running out energy.
Patch Utility	To find the relation between the distance and the utility value assigned to the frontier cells.
Utility Variation	The alteration ratio of the utility value of a patch made by a Marsu choosing that cell.
Alteration Radius	The radius patches that will be altered around a cell chosen by a Marsu.

4 Future Work

As the simulation model is complete and shows a promising approach to having the robots explore a large area that requires multiple charge cycles, the following step is to implement the model into the real Marsu fleet and find out if the robots behave in the same way as their simulated or virtual counterparts. One of the big challenges will be on how to solve the localization problem that comes with creating and assigning the cells, the initial approach to tackle it will have the Marsus create the cells and mark them in the global map with different physical space resolutions (for example squares of 10×10 centimeters).

When the simulation's accuracy is confirmed (taking into account calibration sessions) then the model can be optimized using different approaches such as Genetic Algorithms for choosing the best battery threshold parameters to be able to choose when to go to recharge or when to lower their battery profiles, as well as implementing more advanced approaches to queue theory to improve the handling of the Marsus waiting in line to recharge. All of these parameters would be then be tested in the real Marsu fleet.

References

1. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
2. Burgard, W., Moors, M.,Stachniss, C.: Coordinated Multi-Robot Exploration. IEEE Trans. Robot. **21**, 376–386 (2005)
3. Elomaa, M.: Estimation of unknown node positions of a localization network with a multi-robot system. Espoo, Finland (2010)
4. Gulzar, K.: Impact dampening of a tracked rover. Espoo, Finland (2008)
5. Matusiak, M., Paanajrvi, P., Appelqvist, P., Elomaa, M., Vainio, M., Ylikorpi, T., Halme, A.: A Novel Marsupial robot society: towards long-term autonomy. In: Proceedings of the 9th International Symposium on Distributed Autonomous Robotic Systems (DARS 2008) (2008)
6. Norvig, P., Russell, S.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson International, London (2010)
7. Wilensky, U.: NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston. http://ccl.northwestern.edu/netlogo/ (1999)

Scalable Cooperative Localization with Minimal Sensor Configuration

Xiaotong Shen, Scott Pendleton and Marcelo H. Ang Jr.

Abstract Localization of distributed robots can be improved by fusing the sensor data from each robot collectively in the network. This may allow for each individual robot's sensor configuration to be reduced while maintaining an acceptable level of uncertainty. However, the scalability of a reduced sensor configuration should be carefully considered lest the propagated error become unbounded in large networks of robots. In this paper, we propose a minimal but scalable sensor configuration for a fleet of vehicles localizing on the urban road. The cooperative localization is proven to be scalable if the sensors' data are informative enough. The experimental results justify that pose uncertainty will remain at an acceptable level when the number of robots increases.

Keywords Cooperative localization · Sensor configuration · Scalability

1 Introduction

Cooperative perception extends sensing range beyond line-of-sight and field-of-view by sharing information between agents in the network [7, 10]. In order to merge the information and obtain not only augmented but also consistent observations, the uncertainties of robots' locations should be minimized to a reasonable level. The localization accuracies can be improved knowing that the adjacent robots are sharing the same environment and thus the observations are correlated [5]. The imposed constraints on robots' poses by the relative measurements can also reduce the localization

X. Shen (✉) · S. Pendleton · M.H. Ang Jr.
National University of Singapore, 21 Lower Kent Ridge Road, Singapore, Singapore
e-mail: shen_xiaotong@nus.edu.sg

S. Pendleton
e-mail: scott.pendleton01@nus.edu.sg

M.H. Ang Jr.
e-mail: mpeangh@nus.edu.sg

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_7

uncertainties [3, 13]. Utilizing joint observations and relative measurements are the state-of-the-art techniques for collaborative localization in multi-robot systems [3–5, 8, 11, 13, 15, 17, 19].

The collaborative localization problem can be tackled in a probabilistic manner with data fusion in the probabilistic graph [4, 5]. In the works of Fox et al., the localization uncertainties were further reduced by incorporating sensing data from other robots, though the individual robots equipped with sensors were already able to localize themselves independently without any cooperation. However, the number of required sensors could be reduced for a fleet of robots with sharing sensing information. The minimal number of sensors for a fleet of robots to simultaneously and continually localize themselves remains to be an open question.

Many collaborative localization experiments [3, 5, 8, 11, 17, 19] have been performed indoor or in simulation where features are distinct, sensing error is minor and perception range is small. Madhavan et al. has furthermore conducted outdoor multi-robot localization experiments using slowly moving robots [11]. While promising experimental results have been achieved, they may not extend to outdoor fast moving vehicles. Besides, for the application of on-road autonomous vehicles, each agent may only detect a small number of other vehicles on the road, or even typically just detect the vehicle immediately in front of it, which is a big difference from most indoor scenarios. In this case, the detection graph, whose link is the detection of another robot, is sparse. For an on-road vehicle scenario, the distance between vehicles is also often much larger than in indoor scenarios. The uncertainties induced by the large gaps between vehicles should be considered, especially when driving fast.

The solvability of the localization problem in robot networks has been studied by Dieudonne et al., where they have proven that by using only relative observation the uniqueness of the defined solution could not be guaranteed [3]. The mean positional error decreased with the increased number of robots in [15], but the detection graph was much denser and thus the robots' poses were more correlated to reduce the uncertainty. Will the uncertainties become unbounded as the number of agents and gap distance increase in a sparse detection graph? Bounded uncertainty approaches using bearing constraints are proposed in [17, 19]. However the methods still need each robot to simultaneously observe at least two other robots, which may not hold true in a sparse detection graph.

In this paper we are proposing a minimized sensor configuration for a fleet of vehicles to cooperatively localize themselves on the urban road while driving fast. This research is also motivated by autonomous truck convoy systems [18] and our sensor configuration can greatly reduce the cost of such convoys. We will prove that the uncertainties of using only relative observation will grow unbounded quickly with the increased number of agents and gap distance. With our sensor configuration and proposed algorithm, the scalability is guaranteed and uncertainties will be maintained at a usable level. These are verified by outdoor experiments with three vehicles driving on the road.

The contribution of this paper can be summarized as follows:

- We prove that the uncertainties of cooperative localization using relative observations will grow at least quadratically with the gap distance and linearly with the number of agents.
- If the sensors' data are informative enough, the scalability of our minimal sensor configuration is guaranteed.
- Our proposed algorithm integrates tracking with localization techniques and can minimize the uncertainties to a usable level for a fleet of vehicles on urban road.

The remainder of the paper is organized as follows. Section 2 compares the sensor configurations and probabilistic graphical models. Section 3 introduces our algorithm for cooperative localization. Section 4 provides the experimental results obtained with vehicles driving on urban road. Section 5 concludes this work.

2 Sensor Configurations

2.1 Probabilistic Graphical Models

A typical probabilistic graphical model [20] for localization on a map is shown in Fig. 1. The state S represents the robot pose $(x, y, \theta)^T$ in 2D localization, where x, y represents the position on the map and θ denotes the orientation. In order to localize, a robot usually needs three basic elements, namely the control input U_t, the measurements Z_{mt} of the environment, and the prior map m. Encoders and an inertial measurement unit (IMU) are used to provide the vehicle's odometry and orientation. A LIDAR or camera is often used to get the measurements Z_{mt}. The nodes of the graph which are shaded in darker gray are assumed to be always observed [9].

For multi-robot localization, each robot may detect and track others with some range sensors, such as LIDAR. In the on-road scenario, a vehicle usually can detect and track the vehicle(s) in front of it, namely its leader [7]. The detection of other robots will put constraints on their pose estimations [5]. The prominent difference for on-road vehicles versus other robots is that the detection graph is sparser, as shown in Fig. 2. In Fig. 2a, the detection graph could add five constraints since the

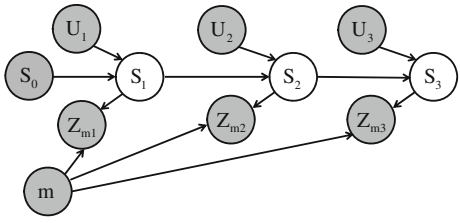


Fig. 1 Typical temporal graphical model for single robot localization

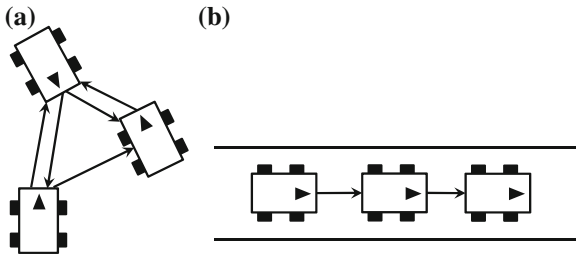


Fig. 2 The detection graph for the multi-robot off-road (**a**) and on-road (**b**) cooperative localization. Each *arrow* represents a robot/vehicle detection (directed from observer to target) and adds a constraint to the localization

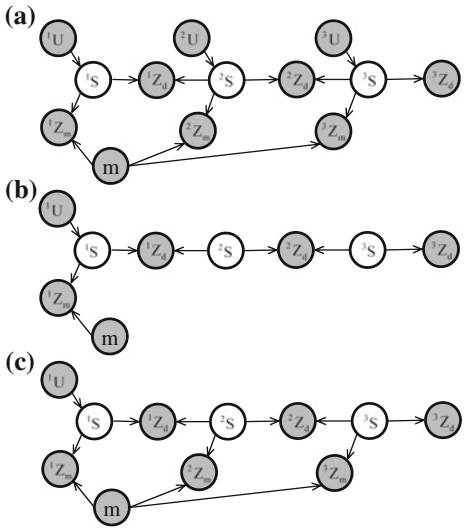


Fig. 3 The comparison of probabilistic graphical models between different sensor configurations for three robots' localization. In (**a**), the control inputs, detection measurements and map measurements are known for each robot. In (**b**), the control inputs and map measurements are unknown except for the first robot and only detection measurement can be used for localization of other robots. In (**c**), both detection measurements and map measurements are known and only the first robot has access to control input

orientations of the sensors face toward each other, while in (b) only two constraints are added since each vehicle could only see its leader vehicle.

Cooperative localization with the full sensor configuration is shown in Fig. 3a, where the ith robot ${}^{i}\mathrm{S}$ is able to localize itself on the map with input ${}^{i}\mathrm{U}$ and ${}^{i}\mathrm{Z}_{m}$ while detecting the $(i+1)$th robot with the measurement ${}^{i}\mathrm{Z}_{d}$. The relative observation ${}^{i}\mathrm{Z}_{d}$, determined by both robots, will correlate both pose estimations.

The sensor configuration could be reduced by taking away either control input U or measurement Z_{m}, or both for some robots, and we need to keep Z_{d} so that the

robots can cooperate with each other. Therefore, there are only three combinations for the reduced sensor configuration if we assume that the sensor configurations on all robots (except the first one) are the same. The reduced sensor configuration is shown in Fig. 3b, c. In Fig. 3b both control input and measurements in map from the two leader vehicles are unknown and the leader's pose ${}^{i+1}\mathrm{S}$ has to be estimated only from relative detection measurement ${}^{i}\mathrm{Z}_d$. We will prove that in this configuration, the uncertainties of cooperative localization will grow unbounded quickly with the increased number of robots and gap distance. If we consider that control input U is given and the map measurements Z_m is unknown, the uncertainties will also be unbounded. The proof for this is similar and is omitted from the paper. In Fig. 3c, the control input is not observable but the measurement of the map is given by the sensor. Essentially, each leader is without an odometry sensor, while the first vehicle (trailing vehicle in the convoy) is still able to localize independently. We will prove that given that the uncertainty of map measurement is small to certain extent, the scalability of cooperative localization under this reduced sensor configuration is guaranteed. Therefore the sensor configuration in Fig. 3c is scalable and minimal.

2.2 Uncertainty Modeling

Assume that the uncertainty of ith robot pose ${}^{i}\mathrm{S} = (x_i, y_i, \theta_i)^T$ can be described by a Gaussian distribution $\mathcal{N}({}^{i}\bar{\mathrm{S}}, {}^{i}\Sigma_p)$, where ${}^{i}\bar{\mathrm{S}} = (\bar{x}_i, \bar{y}_i, \bar{\theta}_i)^T$ is the mean and ${}^{i}\Sigma_p$ is the covariance. The sensor on ith robot can detect the $(i+1)$th robot with some uncertainty. Assume that the sensor is able to detect the relative distance r_i and the bearing angle α_i of the next robot, which is reasonable when using a range sensor such as LIDAR. By utilizing the shape model of the robot, the sensor could also measure the relative orientation ϕ_i. The uncertainty of detection ${}^{i}\mathrm{D} = (r_i, \alpha_i, \phi_i)$ could be modeled as a Gaussian distribution $\mathcal{N}({}^{i}\bar{\mathrm{D}}, {}^{i}\Sigma_d)$, where ${}^{i}\bar{\mathrm{D}} = (\bar{r}_i, \bar{\alpha}_i, \bar{\phi}_i)^T$ is the mean and ${}^{i}\Sigma_d$ is the covariance.

Given the ith robot pose and the detection measurement, the pose of $(i+1)$th robot can be computed as follows:

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} + \begin{pmatrix} r_i cos(\theta_i + \alpha_i) \\ r_i sin(\theta_i + \alpha_i) \\ \phi_i \end{pmatrix}. \tag{1}$$

Because of the nonlinear operations in (1), the distribution of $(i+1)$th robot's pose will be non-Gaussian. Using the first order Taylor expansion to linearize around the mean, the uncertainty could be approximated by a Gaussian distribution. The mean of the $(i+1)$th robot pose can be calculated as follows:

$$\begin{pmatrix} \bar{x}_{i+1} \\ \bar{y}_{i+1} \\ \bar{\theta}_{i+1} \end{pmatrix} = \begin{pmatrix} \bar{x}_i \\ \bar{y}_i \\ \bar{\theta}_i \end{pmatrix} + \begin{pmatrix} \bar{r}_i cos(\bar{\theta}_i + \bar{\alpha}_i) \\ \bar{r}_i sin(\bar{\theta}_i + \bar{\alpha}_i) \\ \bar{\phi}_i \end{pmatrix}. \tag{2}$$

The covariance of the pose can be written as follows:

$$^{i+1}\Sigma_p = G(^i\Sigma_p)G^T + V(^i\Sigma_d)V^T, \tag{3}$$

where G and V are the Jacobian matrices corresponding to ^{i}S and ^{i}D respectively. The G and V can be computed as follows:

$$G = \begin{bmatrix} 1 & 0 & -\bar{r}_i sin(\bar{\theta}_i + \bar{\alpha}_i) \\ 0 & 1 & \bar{r}_i cos(\bar{\theta}_i + \bar{\alpha}_i) \\ 0 & 0 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} cos(\bar{\theta}_i + \bar{\alpha}_i) & -\bar{r}_i sin(\bar{\theta}_i + \bar{\alpha}_i) & 0 \\ sin(\bar{\theta}_i + \bar{\alpha}_i) & \bar{r}_i cos(\bar{\theta}_i + \bar{\alpha}_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

Geometrically, the uncertainty described by the covariance matrix is an ellipsoid in the three dimensional vector space. The determinant of the covariance matrix is proportional to the volume of the ellipsoid [2]. In this paper, we use the determinant of the covariance as the metric of the uncertainty. A large determinant reflects a large uncertainty.

2.3 Scalability Analysis

The scalability of the sensor configuration demonstrates that the uncertainty of each robot's pose remains bounded with an increased number of robots and increased gap distance. Eventually the scalable sensor configuration could accommodate an infinite number of agents in the network. In this analysis, we assume that the state transition is linear and Markovian, and the probability distribution of the pose is Gaussian. We refer to this assumption as Linear-Gauss-Markov model assumption.

We will first prove that under the sensor configuration in Fig. 3b, the uncertainty will grow unbounded quickly.

Lemma 1 $det(^{i+1}\Sigma_p) \geqslant det(^i\Sigma_p) + \bar{r}_i^2 det(^i\Sigma_d)$.

Proof Since $\det(G) = 1$ and $\det(V) = \bar{r}_i$, we have $\det(G(^i\Sigma_p)G^T) = \det(^i\Sigma_p)$ and $\det(V(^i\Sigma_d)V^T) = \bar{r}_i^2 \det(^i\Sigma_d)$. Since $^i\Sigma_p \geq 0$ and $^i\Sigma_d \geq 0$, we have $G(^i\Sigma_p)G^T \geq 0$ and $V(^i\Sigma_d)V^T \geq 0$. According to (3), $\det(^{i+1}\Sigma_p) \geqslant \det(G(^i\Sigma_p)G^T) + \det(V(^i\Sigma_d)V^T) = \det(^i\Sigma_p) + \bar{r}_i^2 \det(^i\Sigma_d)$.

$\det(^i\Sigma_d)$ essentially depicts the uncertainty of detecting the $(i+1)$th robot in the front. The magnitude of $\det(^i\Sigma_d)$ for any robot i should be approximately the same since the sensors used are the same or similar. We assume that the c-th robot has the best accuracy of detection, namely,

$$c = \arg\min_i \det(^i\Sigma_d). \tag{5}$$

Under the sensor configuration in Fig. 3b, the uncertainties would only propagate forward according to (3) since topologically it is a chain.

Theorem 1 *Under the Linear-Gauss-Markov model assumption and with the sensor configuration in Fig. 3b, the uncertainties of cooperative localization will grow at least quadratically with the average gap distance r and linearly with the number of robots n, namely* $det({}^{n}\Sigma_p) \geqslant (n-1)r^2 det({}^{c}\Sigma_d)$.

Proof Iteratively using the result of Lemma 1, we have $\det({}^{n}\Sigma_p) \geqslant \bar{r}_{n-1}^2 \det({}^{n-1}\Sigma_d) + \ldots + \bar{r}_1^2 \det({}^{1}\Sigma_d) + \det({}^{1}\Sigma_p)$. According to (6), $\det({}^{i}\Sigma_d) \geqslant \det({}^{c}\Sigma_d)$ for any $i \leqslant n$. Therefore, $\det({}^{n}\Sigma_p) \geqslant (\bar{r}_{n-1}^2 + \ldots + \bar{r}_1^2)\det({}^{c}\Sigma_d) + \det({}^{1}\Sigma_p)$. Since $\det({}^{1}\Sigma_p) \geqslant 0$ and $r = \dfrac{\sum_{i=1}^{n-1} \bar{r}_i}{n-1}$, we have $\det({}^{n}\Sigma_p) \geqslant (n-1)r^2 \det({}^{c}\Sigma_d)$.

The result of Theorem 1 implies that the cooperative localization is not scalable under the sensor configuration in Fig. 3b. We will prove that under some conditions, the cooperative localization is scalable using the sensor configuration in Fig. 3c.

In the probabilistic graphical model shown in Fig. 3c, numerous cycles would make it difficult to compute the optimal pose estimation while avoiding double counting of the information in the sensor network [12]. We simplify the pose estimation optimization by only fusing the data forward and show that only propagating information forward could make the cooperative localization scalable.

After the ith robot detects the $(i+1)$th robot, the sensor measurement ${}^{i}\Sigma_d$ initializes the location according to (2) and (3). The sensor ${}^{i+1}\Sigma_m$ could search around the initial location and match the features on the map to localize. The uncertainty will be reduced after the fusion of the two measurements. Without relative measurement ${}^{i}\Sigma_d$, it would be very costly and difficult for the $(i+1)$th robot to localize itself on a big map only based on measurement ${}^{i+1}\Sigma_m$, especially when some ambiguity of the environment is prominent. After incorporating the measurement ${}^{i+1}\Sigma_m$, the pose estimation should be improved. We denote pose distribution before incorporation of ${}^{i}\Sigma_m$ as $\mathcal{N}({}^{i}\bar{\mathrm{S}}_b, {}^{i}\Sigma_{pb})$, and after incorporation as $\mathcal{N}({}^{i}\bar{\mathrm{S}}_a, {}^{i}\Sigma_{pa})$. After searching around the ${}^{i}\bar{\mathrm{S}}_b$, the sensor ${}^{i}\Sigma_m$ could infer the pose distributed as $\mathcal{N}({}^{i}\bar{\mathrm{S}}_q, {}^{i}\Sigma_q)$ only relying on measurement ${}^{i}\Sigma_m$. The distribution $\mathcal{N}({}^{i}\bar{\mathrm{S}}_a, {}^{i}\Sigma_{pa})$ is the fusion of $\mathcal{N}({}^{i}\bar{\mathrm{S}}_b, {}^{i}\Sigma_{pb})$ and $\mathcal{N}({}^{i}\bar{\mathrm{S}}_q, {}^{i}\Sigma_q)$. With the Kalman filter fusion strategy, the covariance could be computed as follows:

$$ {}^{i}\Sigma_{pa} = \left(I - {}^{i}\Sigma_{pb}\left({}^{i}\Sigma_{pb} + {}^{i}\Sigma_q\right)^{-1}\right){}^{i}\Sigma_{pb}. \tag{6} $$

Theorem 2 *Under the Linear-Gauss-Markov model assumption, the uncertainty of localization after fusion* ${}^{i}\Sigma_{pb}$ *with* ${}^{i}\Sigma_q$ *will be reduced at least by a factor of* $\dfrac{det({}^{i}\Sigma_q)}{det({}^{i}\Sigma_{pb}) + det({}^{i}\Sigma_q)}$, *namely* $\dfrac{det({}^{i}\Sigma_{pa})}{det({}^{i}\Sigma_{pb})} \leqslant \dfrac{det({}^{i}\Sigma_q)}{det({}^{i}\Sigma_{pb}) + det({}^{i}\Sigma_q)}$.

Proof Since $(A+B)^{-1} = A^{-1} - (I + A^{-1}B)^{-1}A^{-1}BA^{-1}$ if matrices A and $A+B$ are both invertible, we have $\left({}^i\Sigma_{pb} + {}^i\Sigma_q\right)^{-1} = {}^i\Sigma_{pb}^{-1} - (I + {}^i\Sigma_{pb}^{-1\,i}\Sigma_q)^{-1\,i}\Sigma_{pb}^{-1}$ ${}^i\Sigma_q\,{}^i\Sigma_{pb}^{-1}$. Thus, ${}^i\Sigma_{pa} = \left(I - {}^i\Sigma_{pb}\left({}^i\Sigma_{pb} + {}^i\Sigma_q\right)^{-1}\right){}^i\Sigma_{pb} = {}^i\Sigma_{pb}(I + {}^i\Sigma_{pb}^{-1}$ ${}^i\Sigma_q)^{-1\,i}\Sigma_{pb}^{-1\,i}\Sigma_q$.

Therefore, $\det({}^i\Sigma_{pa}) = \det\left({}^i\Sigma_{pb}(I + {}^i\Sigma_{pb}^{-1\,i}\Sigma_q)^{-1\,i}\Sigma_{pb}^{-1\,i}\Sigma_q\right)$

$= \det\left({}^i\Sigma_{pb}\right)\det\left((I + {}^i\Sigma_{pb}^{-1\,i}\Sigma_q)^{-1\,i}\Sigma_{pb}^{-1\,i}\Sigma_q\right) = \det({}^i\Sigma_{pb})\dfrac{\det\left({}^i\Sigma_{pb}^{-1\,i}\Sigma_q\right)}{\det(I + {}^i\Sigma_{pb}^{-1\,i}\Sigma_q)}$

$= \det({}^i\Sigma_{pb})\dfrac{\det\left({}^i\Sigma_q\right)}{\det({}^i\Sigma_{pb} + {}^i\Sigma_q)}$, Since $\det({}^i\Sigma_{pb} + {}^i\Sigma_q) \geqslant \det({}^i\Sigma_{pb}) + \det({}^i\Sigma_q)$,

thus we have $\det({}^i\Sigma_{pa}) \leqslant \det({}^i\Sigma_{pb})\dfrac{\det\left({}^i\Sigma_q\right)}{\det({}^i\Sigma_{pb}) + \det({}^i\Sigma_q)}$.

Essentially, only using the relative measurement for pose estimation will increase the localization uncertainty of the next robot in Lemma 1 and the measurement on the map will reduce the uncertainty in Theorem 2. It is possible that the localization uncertainty of each robot will not increase after the data fusion, which could potentially make the cooperative localization scalable.

Assume that the prescribed uncertainty for any robot in the graph should be less than C_{max}, namely the pose uncertainty $\det({}^i\Sigma_{pa})$ should be less than C_{max} for the localization information to be usable. We will show that if the uncertainty of the sensor measurement $\det({}^i\Sigma_q)$ satisfies the following condition, then C_{max} will be the bound on the localization uncertainty for any robot i.

Theorem 3 *Under the Linear-Gauss-Markov model assumption, if* $det({}^i\Sigma_q) \leqslant \dfrac{C_{max}det({}^i\Sigma_{pb})}{det({}^i\Sigma_{pb}) - C_{max}}$, *then* $det({}^i\Sigma_{pa}) \leqslant C_{max}$ *for any robot in Fig. 3c.*

Proof If $\det({}^i\Sigma_q) \leqslant \dfrac{C_{max}\det({}^i\Sigma_{pb})}{\det({}^i\Sigma_{pb}) - C_{max}}$, then $\det({}^i\Sigma_q)\det({}^i\Sigma_{pb}) \leqslant C_{max}\det({}^i\Sigma_q) +$ $C_{max}\det({}^i\Sigma_{pb})$, thus $\dfrac{\det\left({}^i\Sigma_q\right)\det({}^i\Sigma_{pb})}{\det({}^i\Sigma_{pb}) + \det({}^i\Sigma_q)} \leqslant C_{max}$. With the result of Theorem 2, $\det({}^i\Sigma_{pa}) \leqslant C_{max}$.

The result of Theorem 3 shows that if the sensor measurement on the map ${}^i\Sigma_m$ is informative enough, i.e. the covariance ${}^i\Sigma_q$ is small, then the localization uncertainty could be bounded. The cooperative localization uncertainty is independent of the number robots and their gap distance, which leads to the scalability under sensor configuration shown in Fig. 3c.

In this section, we modeled the uncertainty in the cooperative localization problem and showed that it is not scalable under sensor configuration in Fig. 3b where only relative observations are used. We proved that under the sensor configuration in

Fig. 3c, where odometry information is not given, cooperative localization is scalable if the condition in Theorem 3 is satisfied. Therefore, the reduced sensor configuration in Fig. 3c is minimal for scalable cooperative localization.

3 Cooperative Localization

In this section, we illustrate our optimization method for improving the pose estimation and minimizing the uncertainties of cooperative localization under the minimal sensor configuration.

3.1 Sensor Characteristics

The ith robot could detect and track the $(i+1)$th robot and give an initial pose with some uncertainty. The tracking process usually can estimate the position accurately, but may have difficulties in estimating the orientation. The shape of the robot could be complex, ambiguous and/or only partially observable. The robots will typically maintain a large spatial gap (for road safety), however distant measurements make the orientation estimation more difficult. Nevertheless, the orientation of the $(i+1)$th robot is very critical for localizing the $(i+2)$th robot according to Lemma 1.

The measurement ${}^{i+1}\Sigma_m$ could be used to estimate the orientation accurately, but it would be computationally expensive to search a large area on the map to find the most likely pose. For example, the measurement could be a LIDAR scan of the environment and the alignment of the scan with the map could give an accurate estimation of the orientation if the location of the sensor is roughly known. If no prior knowledge is given, aligning a single scan with the map can be very computationally expensive, especially if the map is huge, ambiguity is prominent and/or the measurement is noisy. Given the initial location, the robot could search locally to find the best pose to maximize the probability of the measurement, which would potentially reduce the uncertainty of the estimation. In summary, tracking and scan matching would compensate each other in pose estimation.

3.2 Temporal Model

The temporal model of cooperative localization is shown in Fig. 4. The state ${}^{i}\mathrm{S}_t$ represents the pose of ith robot at time t. The maps ${}^{i}\mathrm{m}$ are correlated since they are sharing the same environment or are the same. The first robot (trailing vehicle in the convoy) is additionally equipped with odometry and thus can localize itself independently. The first robot can also detect and track the second robot ${}^{2}\mathrm{S}_t$ with

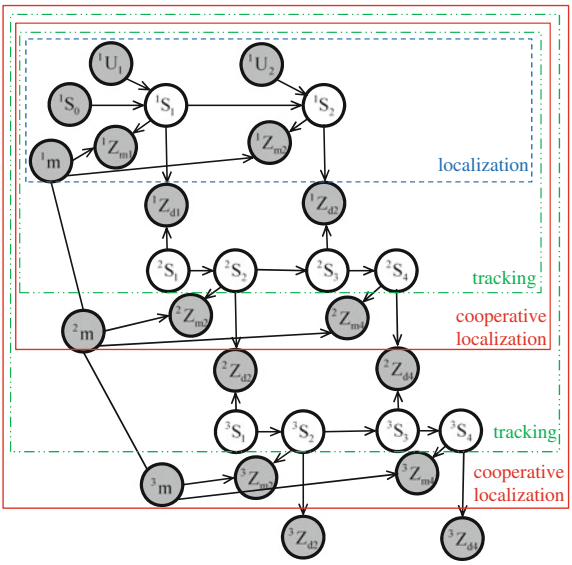


Fig. 4 The temporal model for three robots' cooperative localization. The 1st robot is able to perform independent localization while tracking the 2nd robot. The 2nd robot can perform cooperative localization by fusing the detection measurement from the 1st robot with its local map measurement. The 3rd robot can similarly perform cooperative localization

measurement ${}^1\mathrm{Z}_{dt}$ at time t. The sensor on the second robot can get a measurement of the environment ${}^2\mathrm{Z}_{mt}$ and then can localize itself on the map. Meanwhile, the second robot can detect the third robot with measurement ${}^2\mathrm{Z}_{dt}$. The measurements on each robot may not be synchronized, but we are assuming that the measurements ${}^i\mathrm{Z}_{dt}$ and ${}^i\mathrm{Z}_{mt}$ are from the same sensor and thus the time stamps are the same. For example, a single LIDAR sensor could scan the environment while detecting the moving objects.

3.3 Tracking and Localization

Cooperative localization could be treated as a fusion of tracking and localization. We are using the *Constant Turn Rate and Velocity* (CTRV) model [16] in the Kalman filter for vehicle tracking. When the measurement on the map Z_{mt} comes in, the Bayesian fusion [20] could be performed as follows:

$$p({}^i\mathrm{S}_t|{}^i\mathrm{S}_{bt}, {}^i\mathrm{Z}_{mt}, {}^i\mathrm{m}) = \frac{p({}^i\mathrm{Z}_{mt}|{}^i\mathrm{S}_t, {}^i\mathrm{m})\, p({}^i\mathrm{S}_{bt})}{p({}^i\mathrm{Z}_{mt}|{}^i\mathrm{S}_{bt}, {}^i\mathrm{m})}, \tag{7}$$

where the state ${}^i\mathrm{S}_{bt}$ is the predicted robot pose by the Kalman tracker and state ${}^i\mathrm{S}_t$ is the pose after fusing measurement ${}^i\mathrm{Z}_{mt}$. The probability $p({}^i\mathrm{S}_{bt})$ could be given by the Kalman tracker. The measurement probability $p({}^i\mathrm{Z}_{mt}|{}^i\mathrm{S}_t, {}^i\mathrm{m})$ could be evaluated by the likelihood model [20] since the prior map ${}^i\mathrm{m}$ is given.

Algorithm 1: Cooperative Localization Algorithm

input : ${}^{i}\bar{\mathrm{S}}_{p(t-1)}, {}^{i}\Sigma_{p(t-1)}, {}^{i}\mathrm{Z}_{mt}, {}^{i-1}\mathrm{Z}_{d(t+1)}, {}^{i-1}\bar{\mathrm{S}}_{p(t+1)}, {}^{i-1}\Sigma_{p(t+1)}$

output: ${}^{i}\bar{\mathrm{S}}_{p(t+1)}, {}^{i}\Sigma_{p(t+1)}$

//localization

$({}^{i}\bar{\mathrm{S}}_{bt}, {}^{i}\Sigma_{bt})$= `KalmanPrediction` $({}^{i}\bar{S}_{p(t-1)}, {}^{i}\Sigma_{p(t-1)})$

for *j= 1 ,...,N* **do**

draw sample ${}^{i}_{j}\mathrm{S}_{bt} \sim \mathcal{N}({}^{i}\bar{\mathrm{S}}_{pt}, {}^{i}\Sigma_{pt})$

${}^{i}_{j}\hat{\mathrm{S}}_{t} = \arg\max_{{}^{i}\mathrm{S}_{t}} p({}^{i}\mathrm{Z}_{mt}|{}^{i}\mathrm{S}_{t}, {}^{i}\mathrm{m}, {}^{i}_{j}\mathrm{S}_{bt}), \quad {}^{i}_{j}w = p({}^{i}_{j}\mathrm{S}_{bt}|{}^{i}\bar{\mathrm{S}}_{pt}, {}^{i}\Sigma_{pt})\, p({}^{i}\mathrm{Z}_{mt}|{}^{i}\mathrm{m}, {}^{i}_{j}\hat{\mathrm{S}}_{t})$

end

$l = \arg\max_{j} {}^{i}_{j}w, \quad {}^{i}\mathrm{S}_{qt} = {}^{i}_{l}\hat{\mathrm{S}}_{t}$ //search for the most likely mode

for *k= 1 ,...,K* **do**

draw sample ${}^{i}_{k}\mathrm{S}_{qt} \sim \{{}^{i}_{k}\mathrm{S}_{qt} | \ |{}^{i}_{k}\mathrm{S}_{qt} - {}^{i}\mathrm{S}_{qt}| < \Delta\}$ //sample around the mode

end

${}^{i}\bar{\mathrm{S}}_{qt} = (0,0,0)^{T}, \eta = 0$

for *k=1,...,K* **do**

${}^{i}\bar{\mathrm{S}}_{qt} = {}^{i}\bar{\mathrm{S}}_{qt} + {}^{i}_{k}\mathrm{S}_{qt} \cdot p({}^{i}\mathrm{Z}_{mt}|{}^{i}\mathrm{m}, {}^{i}_{k}\mathrm{S}_{qt}), \quad \eta = \eta + p({}^{i}\mathrm{Z}_{mt}|{}^{i}\mathrm{m}, {}^{i}_{k}\mathrm{S}_{qt})$

end

${}^{i}\bar{\mathrm{S}}_{qt} = {}^{i}\bar{\mathrm{S}}_{qt}/\eta, \quad {}^{i}\Sigma_{qt} = \mathbf{0}$

for *k=1,...,K* **do**

${}^{i}\Sigma_{qt} = {}^{i}\Sigma_{qt} + ({}^{i}_{k}\mathrm{S}_{qt} - {}^{i}\bar{\mathrm{S}}_{qt})({}^{i}_{k}\mathrm{S}_{qt} - {}^{i}\bar{\mathrm{S}}_{qt})^{T} \cdot p({}^{i}\mathrm{Z}_{mt}|{}^{i}\mathrm{m}, {}^{i}_{k}\mathrm{S}_{qt})$

end

${}^{i}\Sigma_{qt} = {}^{i}\Sigma_{qt}/\eta$

$({}^{i}\bar{\mathrm{S}}_{pt}, {}^{i}\Sigma_{pt})$= `KalmanFusion` $({}^{i}\bar{S}_{qt}, {}^{i}\Sigma_{qt}, {}^{i}\bar{S}_{bt}, {}^{i}\Sigma_{bt})$

//tracking

$({}^{i}\bar{\mathrm{S}}_{b(t+1)}, {}^{i}\Sigma_{b(t+1)})$= `KalmanPrediction` $({}^{i}\bar{S}_{pt}, {}^{i}\Sigma_{pt})$

$({}^{i-1}\bar{\mathrm{D}}_{t+1}, {}^{i-1}\Sigma_{d(t+1)})$ = `Detection` $({}^{i-1}Z_{d(t+1)})$

${}^{i}\bar{\mathrm{S}}_{pd(t+1)} = {}^{i-1}\bar{\mathrm{S}}_{p(t+1)} \oplus {}^{i-1}\bar{\mathrm{D}}_{t+1}, \quad {}^{i}\Sigma_{pd(t+1)} = G({}^{i-1}\Sigma_{p(t+1)})G^{T} + V({}^{i-1}\Sigma_{d(t+1)})V^{T}$

$({}^{i}\bar{\mathrm{S}}_{p(t+1)}, {}^{i}\Sigma_{p(t+1)})$=`KalmanFusion` $({}^{i}\bar{S}_{pd(t+1)}, {}^{i}\Sigma_{pd(t+1)}, {}^{i}\bar{S}_{b(t+1)}, {}^{i}\Sigma_{b(t+1)})$

return ${}^{i}\bar{\mathrm{S}}_{p(t+1)}, {}^{i}\Sigma_{p(t+1)}$

The improved proposal distribution technique [6] could improve the localization. We sample N particles ${}^{i}_{j}\mathrm{S}_{bt}$ from the predicted distribution $p({}^{i}\mathrm{S}_{bt})$ but adopt the optimization technique to search for the maxima of the observation likelihood, knowing that it is most likely to have only a single maxima [6]. The scan matching will determine the meaningful area of the observation ${}^{i}\mathrm{Z}_{mt}$ likelihood function and the particles will iteratively shift to reach the most probable pose even if the predicted distribution is inaccurate. We use the " vasco" scan matcher which is part of the Carnegie Mellon Robot Navigation Toolkit (CARMEN) [14]. This scan matcher uses a gradient descent technique to find the most likely pose by matching the observation ${}^{i}\mathrm{Z}_{mt}$ against the map ${}^{i}\mathrm{m}$,

$$ {}^{i}_{j}\hat{\mathrm{S}}_t = \arg\max_{{}^{i}\mathrm{S}_t} p({}^{i}\mathrm{Z}_{mt} | {}^{i}\mathrm{S}_t, {}^{i}\mathrm{m}, {}^{i}_{j}\mathrm{S}_{bt}),\ j = 1, ..., N, \tag{8} $$

where ${}^{i}_{j}\hat{\mathrm{S}}_t$ is the optimized pose with the initial guess ${}^{i}_{j}\mathrm{S}_{bt}$ and measurement ${}^{i}\mathrm{Z}_{mt}$. Among all the N particles, we choose the most likely pose when fusing the proposal distribution and the observation,

$$ {}^{i}\mathrm{S}_{qt} = \arg\max_{{}^{i}_{j}\hat{\mathrm{S}}_t} p({}^{i}_{j}\hat{\mathrm{S}}_t | {}^{i}\mathrm{Z}_{mt}, {}^{i}\mathrm{m}, {}^{i}_{j}\mathrm{S}_{bt}),\ j = 1, ..., N, \tag{9} $$

where ${}^{i}\mathrm{S}_{qt}$ is the best mode of the poses. To obtain the distribution of $\mathcal{N}({}^{i}\bar{\mathrm{S}}_q, {}^{i}\Sigma_q)$, we can sample K particles around the ${}^{i}\mathrm{S}_{qt}$ within the interval Δ and weigh them by the observation likelihood. The details are summarized in Algorithm 1. The `KalmanPrediction` is to use the CTRV model to predict the future pose and the `KalmanFusion` is to use Kalman equation to update the belief with the measurement input [1]. The `Detection` in Algorithm 1 is to extract the next robot pose with the measurement ${}^{i-1}\mathrm{Z}_{dt}$. The operator $\oplus$ represents the transform from the detection to the next robot's pose estimation ${}^{i}\bar{\mathrm{S}}_{pd(t+1)}$, which is shown in (2).

4 Experimental Results

In this section, we provide experimental results to justify the theorems and evaluate the algorithm with three vehicles driving on the urban road.

4.1 Experiment Setup

In these experiments, we used three vehicles driving on the urban road to perform the cooperative localization. The first robot (trailing vehicle in the convoy) is a self-driving vehicle which can autonomously drive on the urban road with centimeter-level localization accuracy. The vehicle is equipped with one 2D LIDAR, two encoders, one IMU, one camera and two computers with wireless interface 802.11n.

The other two vehicles are only mounted with one 2D LIDAR individually for both detecting the vehicle in the front and localizing itself. One webcam and a laptop with wireless interface are utilized for visualization and processing the data respectively. The whole localization package is portable and easy to mount without too much mechanical work. The reduced sensors are one IMU and two encoders, which require mechanical connection with vehicles. The experiment was performed on an urban road with average speed 3 m/s. A snapshot of the cooperative localization experiment is shown in Fig. 5. A video showing the cooperative localization process can be found at http://youtu.be/7uuQPgw1rmg.

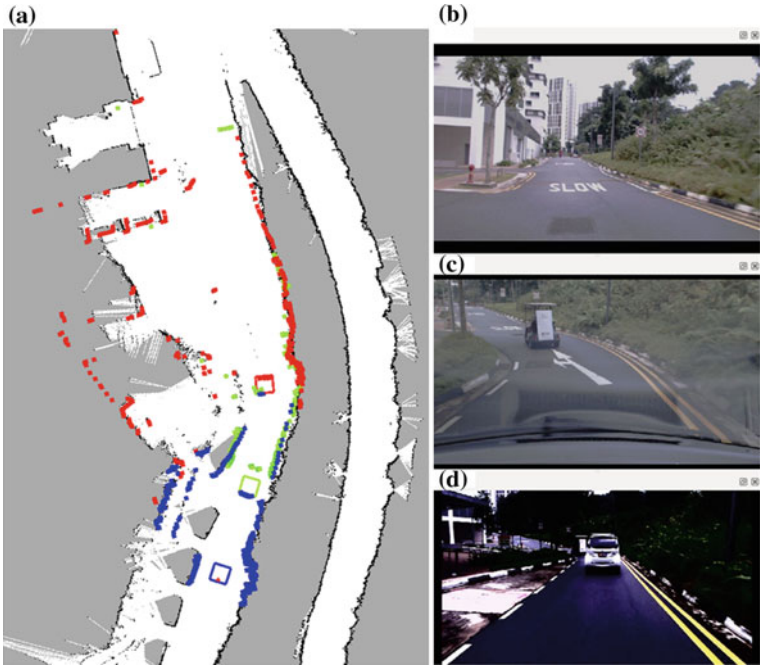


Fig. 5 A snapshot of the cooperative localization on the urban road. The three vehicles are shown in *red*/*green*/*blue boxes* in (**a**). The LIDAR scans from each vehicle are shown in *dots* using corresponding colors on the prior occupancy grid map. The images from three vehicles are shown in (**b**–**d**) which are from the vehicles in *red*/*green*/*blue* respectively. The second and third vehicle can be seen from the first vehicle's perspective in (**d**)

4.2 Evaluation and Analysis

In Fig. 5a, the third vehicle (depicted as the red box) in the front was making a turn. The LIDAR measurement from the second vehicle, in green color, could only detect the corner of the third vehicle. Even though the initial position from the detection was inaccurate, its LIDAR scan managed to correct its pose by matching the scan with the prior occupancy grid map. The pose estimation of the second and third vehicle are improved by the scan matching in the cooperative localization algorithm.

Table 1 shows the quantitative result of pose estimation using only the Relative Observations (RO) method and using the Cooperative Localization (CL) algorithm. Because we have no other means to obtain the ground truth of the poses and the first vehicle could localize very accurately, we are assuming that the pose trajectory of the first vehicle is the ground truth. Both position and orientation error are calculated relative to the trajectory of the first vehicle. Since the trajectories are not exactly the same in reality, the standard deviation and maximum error should be more suitable for analyzing the accuracy. In Table 1, the standard deviation of 2nd and 3rd vehicles'

Table 1 Accuracy of pose estimation

	Position error (m)				Orientation error (°)			
	2nd Vehicle		3rd Vehicle		2nd Vehicle		3rd Vehicle	
	RO	CL	RO	CL	RO	CL	RO	CL
Average	0.32	0.34	0.95	0.72	0.7	1.7	−0.5	−0.6
Std Dev	0.3	0.3	0.72	0.35	3.6	2.7	5.25	2.64
Max	1.31	1.39	4.12	1.52	9.7	9.1	24.3	6.29

poses are about the same using the CL algorithm while that of 3rd vehicle's pose is about twice as large as that of 2nd vehicle's when using the RO method. The maximum position error when using RO is more than two times larger than the CL algorithm while the maximum orientation error is about three times larger. Since the standard deviation and maximum error of 2nd and 3rd vehicles' pose estimation is approximately equivalent when using CL, this implies that the sensor configuration using single 2D LIDAR for both tracking and localization is scalable.

4.3 Error Correlation

In Fig. 6c, the pose error of the 2nd vehicle induced approximately three times larger pose error for the 3rd vehicle. The pose estimation error is highly correlated between

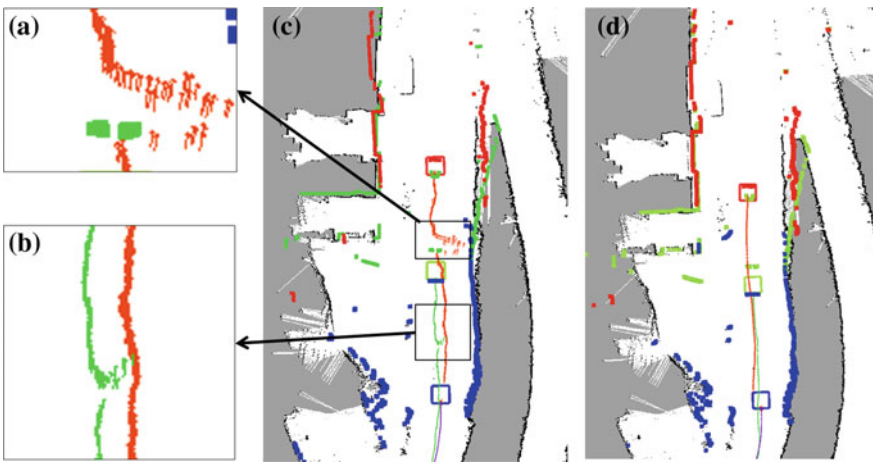


Fig. 6 The error amplification effect when using relative measurement only. The mean pose trajectory of the 3rd vehicle is shown in *red* in (**a**) and that of the 2nd vehicle is shown in *green* in (**b**). The trajectories of the three vehicles are plotted in *red*/*green*/*blue* colors using RO (**c**) method and CL (**d**) algorithm

the 2nd and the 3rd vehicle when using RO. Conversely, the trajectories of both 2nd and 3rd vehicle are very smooth in Fig. 6d, where the pose error was minimized locally to avoid the huge error propagation. By matching the scan with the prior occupancy grid map, the orientation error was reduced such that the initial position of next vehicle given by the detection measurement was accurate enough for next vehicle's localization, which makes the system scalable.

5 Conclusion

In this paper, we proved that using only relative observations for cooperative localization is not scalable. Given that the measurements on the map are informative enough, the cooperative localization under our minimal sensor configuration was proven to be scalable. We proposed the cooperative localization algorithm for a fleet of vehicles localizing on the urban road, which integrates both tracking and localization techniques. The experimental results showed that the pose estimation uncertainties for all three vehicles were minimized to an acceptable level by fusing the detection and map measurements. The uncertainties were reduced to an extent such that the cooperative localization is scalable.

Acknowledgments This research was supported by the National Research Foundation (NRF) Singapore through the Campus for Research Excellence And Technological Enterprise (CREATE) and the Singapore MIT Alliance for Research and Technology's (FM IRG) research programme, in addition to the partnership with the Defence Science Organisation (DSO). We are grateful for their support.

References

1. Blackrnan, S., House, A.: Design and Analysis of Modern Tracking Systems. Artech House, Boston (1999)
2. Butler, R., Davies, P., Jhun, M., et al.: Asymptotics for the minimum covariance determinant estimator. Ann. Stat. **21**(3), 1385–1400 (1993)
3. Dieudonné, Y., Labbani-Igbida, O., Petit, F.: On the solvability of the localization problem in robot networks. In: 2008 IEEE International Conference on Robotics and Automation (ICRA), pp 480–485. IEEE (2008)
4. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: Collaborative multi-robot localization. In: Mustererkennung 1999, Springer, pp. 15–26 (1999)
5. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. Auton. Robots **8**(3), 325–344 (2000)
6. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Robot. **23**(1), 34–46 (2007)
7. Kim, S.W., Chong, Z.J., Qin, B., Shen, X., Cheng, Z., Liu, W., Ang, M.H.: Cooperative perception for autonomous vehicle control on the road: Motivation and experimental results. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5059–5066. IEEE (2013)

8. Knuth, J., Barooah, P.: Distributed collaborative localization of multiple vehicles from relative pose measurements. In: 47th Annual Allerton Conference on Communication, Control, and Computing, 2009. Allerton 2009, pp. 314–321. IEEE (2009)
9. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT Press, Cambridge (2009)
10. Li, H., Nashashibi, F.: Multi-vehicle cooperative perception and augmented reality for driver assistance: a possibility to 'see' through front vehicle. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 242–247. IEEE (2011)
11. Madhavan, R., Fregene, K., Parker, L.E.: Distributed heterogeneous outdoor multi-robot localization. In: 2002 IEEE International Conference on Robotics and Automation (ICRA), vol. 1, pp. 374–381. IEEE (2002)
12. Manyika, J., Durrant-Whyte, H.: Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach. Prentice Hall PTR, Englewood Cliffs (1995)
13. Martinelli, A., Pont, F., Siegwart, R.: Multi-robot localization using relative observations. In: 2005 IEEE International Conference on Robotics and Automation (ICRA), pp. 2797–2802 (2005)
14. Montemerlo, D., Roy, N., Thrun, S.: Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (carmen) toolkit. In: 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2436–2441. IEEE (2003)
15. Rekleitis, I.M., Dudek, G., Milios, E.E.: Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In: 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2690–2695. IEEE (2002)
16. Schubert, R., Richter, E., Wanielik, G.: Comparison and evaluation of advanced motion models for vehicle tracking. In: 2008 11th International Conference on Information Fusion, pp. 1–6. IEEE (2008)
17. Spletzer, J.R., Taylor, C.J.: A bounded uncertainty approach to multi-robot localization. In: 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 2, pp. 1258–1265. IEEE (2003)
18. Switkes, J.P., Gerdes, J.C., Berdichevsky, G., Berdichevsky, E.: Systems and methods for semi-autonomous vehicular convoys. US Patent App. 13/542,622 (2012)
19. Taylor, C., Spletzer, J.: A bounded uncertainty approach to cooperative localization using relative bearing constraints. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2500–2506 (2007)
20. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)

Towards Cooperative Localization in Robotic Swarms

Anderson G. Pires, Douglas G. Macharet and Luiz Chaimowicz

Abstract Cooperative localization allows groups of robots to improve their overall localization by sharing position estimates within the team. In spite of being a well studied problem, very few works deal with the increased complexity when a large number of robots is used, as is the case in robotic swarms. In this paper, we present a characterization and analysis of the cooperative localization problem for robotic swarms. We use a decentralized cooperative mechanism in which robots take turns as dynamic landmarks providing information to their teammates. We perform several simulations and analyze the influence of these dynamic landmarks in the localization. More specifically, we study the impact of the number of robots in the localization and how the choice of landmarks affects the results.

Keywords Cooperative localization · Cooperative mobile robots · Swarm robotics

1 Introduction

The localization problem is one of the most fundamental in mobile robotics. It generally consists in estimating the robot pose relative to a reference frame in the environment. When robots are equipped with exteroceptive sensors (such as laser range finders) and a set of known landmarks or a map of the environment is available,

A.G. Pires (✉) · D.G. Macharet · L. Chaimowicz
Computer Vision and Robotics Laboratory (VeRLab), Computer Science Department – Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, MG, Brazil
e-mail: anderson@leopoldina.cefetmg.br

D.G. Macharet
e-mail: doug@dcc.ufmg.br

L. Chaimowicz
e-mail: chaimo@dcc.ufmg.br

A.G. Pires
Computer and Mechanics Department – Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Leopoldina, MG, Brazil

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_8

localization is relatively simple. This is also true for outdoor robots equipped with a good GPS, which can provide position estimates in a global reference frame. But in more general settings, in which GPS is not available and the robot has no knowledge about the environment, robots have to rely on dead reckoning methods that compute new pose estimates from previous ones. Unfortunately, these methods are subjected to accumulated errors when traveling long distances, which lead to uncertainties that may compromise the quality of the localization.

In multi-robot teams, individual localization estimates can be corrected based on the teammates' positions instead of landmarks in the environment. This is one of the benefits of cooperative robotics, which allows robots to share responsibilities and exchange information to better accomplish tasks. The pose belief adjustment occurs by means of information exchange, which generally happens when the robots meet each other (i.e. there are robots within the communication range). Multi-robot systems employing this technique, commonly denoted Cooperative Localization (CL), have less dependence on the availability of global localization information. Consequently, this kind of system can be used to explore unknown areas or scenarios where global localization is not always available.

These advantages can be leveraged with the use of large groups of robots, which usually present increased flexibility and robustness. Generally called *Robotic Swarms*, these systems employ a large number of simpler agents to perform different types of tasks, acting in a completely decentralized fashion. As will be discussed in the next section, most of the CL methods have focused on the use of a small number of robots since the complexity in terms of coordination and information exchange increases with the number of robots. Thus, the problem of CL in robotic swarms has not been fully explored and presents relevant questions to be investigated.

In this paper, we present a characterization and analysis of the cooperative localization problem considering a swarm of robots. We use a decentralized method, in which the main idea is to have some robots in the swarm acting as dynamic landmarks and providing a localization structure to the group. More specifically, we have the members of the group alternately working as localization providers. These members, acting as beacons, publish localization information in their vicinities to allow their neighbors to adjust their localization beliefs. In this context, we perform an extensive series of simulations and analyze how the number of dynamic landmarks and their choice may impact the localization in a robotic swarm.

The remainder of this paper is structured as follows. A review on the CL literature is presented in Sect. 2. The methodology is presented in Sect. 3, which initially describes the theoretical formalizations (Sect. 3.1), followed by the cooperative swarm localization method (Sect. 3.2), and the swarm motion strategy (Sect. 3.3) used to move the group as a unit. Experiments and statistical analysis are presented in Sect. 4. Finally, Sect. 5 brings the conclusions and directions for future work.

2 Related Work

One of the first works that use robots as landmarks to perform cooperative localization is [6], in which a group of robots, with awareness of its initial localization, is divided in two subgroups with alternating motion and roles. At each time-step, one group is in motion while the other remains static to serve as landmark. After the motion, the robots update their localization beliefs by using relative observations and then remain stationary acting as landmarks to the other group. Despite the good results shown in real applications [5], the need of a centralized entity that controls the actions of all robots and estimates their beliefs compromises the robustness and scalability of the method.

Another seminal work is [7], in which the concept of cooperative localization is employed in a task related to mapping. Two robots are equipped with sensors that allow them to track each other. The coordination mechanism permits them to divide the environment by using spatial decomposition, such that at any single time one robot acting as landmark is positioned in a corner of the environment, while the other spans the perimeter maintaining visual contact. Therefore, the regions of the entire free space are covered and a dual graph is generated, which can be used to posterior exploration of the area.

A more general approach to the CL problem is presented in [10]. The method is based on the generation of a *joint* estimation of the robots' pose in a group, which is computed using an Extended Kalman Filter (EKF) [4]. Both centralized and decentralized methods were applied to generate the joint estimation. In the decentralized approach, each robot performs the prediction step of the filter individually while the update step is performed by exchanging information with others via communication and exteroceptive sensors. The localization interdependence is considered and its representation (cross correlation terms) are stored by all robots and explicitly propagated to the teammates. Using these terms, a robot can estimate its pose by considering the shared knowledge associated with previous meetings. In spite of having the best estimate as a consequence of the use of localization interdependence, this strategy has the disadvantage of requiring a previous knowledge about the group size and presents a complexity that increases quadratically with the number of robots, precluding its use for large groups of robots, such as swarms.

To deal with this drawback, other works have proposed approximate strategies to perform the belief update. These approaches use only part of the group to calculate the robot's estimate. In a recent approximate approach [1], the belief update is performed by using the Covariance Intersection Algorithm (CI) [3], which is a consistent method to fuse estimates of a same quantity with unknown cross-correlation terms. This approach allows each robot to maintain only its own state-covariance estimate with a cost to generate a new estimate of $O(n)$.

Some works have investigated different strategies to increase the quality of localization. For instance, [2, 14, 15] explore the motion mechanism of the group. Tully et al. [14], for example, presents a leap-frog motion technique that has been designed to aid localization for a team of three robots that move alternatively. The results show

that this motion strategy outperforms the optimal formation-based path. A method based on leader-following is presented in [16], in which they explore the formation to generate optimal motion strategies. Two robots are employed and the simulation results present better localization accuracy in comparison with the other formation methods used.

However, just a few works have studied the influence of the group size in relation to the quality of the group localization. In [11], a theoretical analysis relates the effect of the number of robots and the error accumulation. In this analysis, the continuous exchange of localization information among the robots is considered and each robot uses sensors of limited accuracy to provide absolute orientation. It is shown that the uncertainty growth is inversely proportional to the number of robots and the rate of growth depends only on this number and the uncertainty of proprioceptive sensors. In [8], this aspect is evaluated together with the type of measurement used. Although the results have shown that the increase in the number of robots contributes to the quality of localization, relevant details have not become explicit, such as the number of robots used as landmarks, which prevents a more detailed analysis. In a recent work [12], the influence of the group size is studied in simulated experiments using groups from two to five robots. A centralized EKF sequentially performs the localization estimate of the group using the data sent by all robots. Because of the restricted scope of the experiments, the results cannot be generalized.

Thus, in spite of the different studies in cooperative localization, its use in robotic swarms is still incipient. In special, the study of scalability issues in this context is a relevant problem that we consider suitable for investigation.

3 Methodology

In this work, we consider a large group of robots (swarm) that moves in a cohesive manner. As in [6, 7], the swarm is divided into two subgroups that move in a mutually exclusive way. One group remains stationary broadcasting their pose information while the robots of the other group move using proprioceptive sensors to estimate their pose. After moving for a certain amount of time, each of these robots updates its belief by using some of the stationary robots in its neighborhood as landmarks. The process is completely decentralized: each robot estimates its distance and orientation to the landmarks, and use the pose information disseminated by them to correct its pose. After this, robots exchange roles: the stationary group starts moving while the robots of the other group become stationary landmarks.

Similarly to [1], we consider that robots are able to identify and measure relative ranges and bearings to their neighbors and exchange information with them. Also, robots are equipped with proprioceptive sensors that allow them to measure their own motion. Since we are using holonomic robots, we do not consider the robot orientation. We assume that all sensor measurements are subjected to white Gaussian noise, but communication is performed without errors.

3.1 Theoretical Formalization

Let us consider the scenario where a swarm $\boldsymbol{\mathcal{R}} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_\eta\}$ of η holonomic robots must navigate in a static 2D environment. Let $\mathbf{p}_k^i = [x_k^i \; y_k^i]^\top$ be the vector at time-step k that represents the true position of the ith robot ($\mathcal{R}_i$) in a common global frame W, and $\mathbf{u}_k^i = [vx_k^i \; vy_k^i]^\top$ the vector that represents its control action in the same time-step, in which vx_k^i and vy_k^i stand for the input velocities in x and y directions, respectively.

The state $\mathbf{x}_k^i$ of the robot $\mathcal{R}_i$ at time-step k is equal to its position, *i.e.* $\mathbf{x}_k^i = \mathbf{p}_k^i = [x_k^i \; y_k^i]^\top$, and its discrete-time motion model is expressed by:

$$\begin{aligned} \mathbf{x}_{k+1}^i &= f(\mathbf{x}_k^i, \mathbf{u}_k^i), \qquad i = 1, \ldots, \eta \\ &= \begin{bmatrix} x_k^i + vx_k^i \Delta k \\ y_k^i + vy_k^i \Delta k \end{bmatrix}. \end{aligned} \tag{1}$$

A neighborhood $\mathcal{N}$ consists of a circular region of radius $\boldsymbol{\tau}$ around the current position of a robot. Thus, we can define $\boldsymbol{\mathcal{N}} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_\eta\}$ as the set of calculated neighborhoods, all with the same radius. As mentioned, we assume that $\mathcal{R}_i$ can exchange information and measure relative range ρ and bearing ϕ of all robots inside its neighborhood $\mathcal{N}_i$. Moreover, it is assumed that robots inside a neighborhood $\mathcal{N}_i$ can be uniquely identified by the exteroceptive sensor of robot $\mathcal{R}_i$.

The true range and bearing taken by robot i of a robot j at time-step k are respectively denoted by $\rho_k^{i,j}$ and $\phi_k^{i,j}$.[1] Thus, the true range $\rho_k^{i,j}$ and bearing $\phi_k^{i,j}$ taken at time-step k by robot i of robot j, is given by $h(\mathbf{x}_k^i, \mathbf{x}_k^j)$, where

$$h(\mathbf{x}_k^i, \mathbf{x}_k^j) = \begin{bmatrix} \rho_k^{i,j} \\ \phi_k^{i,j} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k^j - x_k^i)^2 + (y_k^j - y_k^i)^2} \\ \text{atan2}(y_k^j - y_k^i, x_k^j - x_k^i) \end{bmatrix}. \tag{2}$$

The measurement model at time-step $k+1$, when $\mathcal{R}_i$ gets a relative position measurement of $\mathcal{R}_j$, $\mathbf{z}_{k+1}^{i,j} = [\hat{\rho}_{k+1}^{i,j} \; \hat{\phi}_{k+1}^{i,j}]$, $i, j = 1, \ldots, \eta$, $i \neq j$, $j \in \mathcal{N}_i$, is given by

$$\mathbf{z}_{k+1}^{i,j} = h(\mathbf{x}_{k+1}^i, \mathbf{x}_{k+1}^j) + \mathbf{n}_{k+1}^{i,j}, \tag{3}$$

where $\mathbf{n}_{k+1}^{i,j}$ is the zero-mean white Gaussian measurement noise with covariance $\mathbf{R}_{k+1}^{i,j}$ added to the true relative measurements given by $h(\mathbf{x}_{k+1}^i, \mathbf{x}_{k+1}^j)$.

[1] We use the notation $*_k^{i,j}$ to express that a certain value associated with robot i was obtained using information and/or measurements from robot j at time-step k.

3.2 Cooperative Swarm Localization

As described earlier, this work uses a swarm $\mathcal{R}$ of η holonomic robots. The swarm is divided in two subgroups, and their motions are coordinated such that the subgroups move as units in a mutually exclusive way (see Sect. 3.3). Individually, each robot i maintains only its own state estimate $\hat{\mathbf{x}}_k^i$ and the respective covariance $\mathbf{P}_k^i$, due to the costs of processing and communication when cooperatively localizing a large group of robots. In this work we do not address the cross-correlation terms [10], and propose an approximate decentralized algorithm for CL.

A robot $\mathcal{R}_j$ acting as landmark continually broadcasts messages with its state and covariance to its neighbors. After its motion, a robot $\mathcal{R}_i$ trying to localize itself, processes the relative range and bearing measurement $\mathbf{z}_{k+1}^{i,j}$ together with the information received from robot $\mathcal{R}_j$. Using these data and its own predicted state $\hat{\mathbf{x}}_{k+1|k}^i$[2] and covariance $\mathbf{P}_{k+1|k}^i$ estimates, the robot processes the new state $\hat{\mathbf{x}}_{k+1|k+1}^i$ and covariance $\mathbf{P}_{k+1|k+1}^i$ estimates. This procedure is repeated incrementally for each landmark in order to improve the localization estimates. The mathematical details of this procedure is presented as follows.

The discrete-time motion model described in (1) is used to propagate the state of the robot $\mathcal{R}_i$ as:

$$\hat{\mathbf{x}}_{k+1|k}^i = f(\hat{\mathbf{x}}_{k|k}^i, \hat{\mathbf{u}}_k^i), \qquad i = 1, \ldots, \eta. \tag{4}$$

The robot's state is updated according to a linear function f that considers the previous state $\hat{\mathbf{x}}_k^i$ and an input $\hat{\mathbf{u}}_k^i = \mathbf{u}_k^i + \mathbf{w}_k^i = [\widehat{vx}_k^i \ \widehat{vy}_k^i]^\top$, which is basically the commanded velocities $\mathbf{u}_k^i$ augmented with additive zero-mean white Gaussian noise $\mathbf{w}_k^i$, with covariance $\mathbf{Q}_k^i$. During the motion, each robot individually evolves this model with time-steps of length Δk.

Using an EKF [4], the respective covariance propagation for $\mathcal{R}_i$ is given by:

$$\mathbf{P}_{k+1|k}^i = \Phi_k^i \mathbf{P}_{k|k}^i (\Phi_k^i)^\top + \mathbf{G}_k^i \mathbf{Q}_k^i (\mathbf{G}_k^i)^\top, \tag{5}$$

where Φ_k^i is a 2×2 identity matrix ($\mathbf{I}_2$) and $\mathbf{G}_k^i$ is this same matrix multiplied by the time-step Δk.

Thus, when $\mathcal{R}_i$ receives the message with the state and covariance estimates from robot j and obtains a relative measurement $\mathbf{z}_{k+1}^{i,j}$ of it, $\mathcal{R}_i$ can generate an estimate of its state as if such estimate had been calculated by the robot $\mathcal{R}_j$. The following equation illustrates this process:

$$\hat{\mathbf{x}}_{k+1}^{i,j} = \hat{\mathbf{x}}_{k+1|k+1}^j - g(\mathbf{z}_{k+1}^{i,j}), \tag{6}$$

[2] Notation is similar to [1], where $\hat{\mathbf{y}}_{l|m}$ denotes the estimate of the random variable $\mathbf{y}$ at time-step l, given the measurements up to time-step m.

where

$$g(\mathbf{z}_{k+1}^{i,j}) = \begin{bmatrix} \hat{\rho}_{k+1}^{i,j}\cos(\hat{\phi}_{k+1}^{i,j}) \\ \hat{\rho}_{k+1}^{i,j}\sin(\hat{\phi}_{k+1}^{i,j}) \end{bmatrix}.$$

As described in [13], the uncertainty $\mathbf{R}_{k+1}^{i,j}$ tied to the measurement $\mathbf{z}_{k+1}^{i,j}$ can be converted to the common global frame by the means of the jacobian $\mathbf{J}_{k+1}^{i,j}$, as follows:

$$\begin{aligned} \mathbf{J}_{k+1}^{i,j} &= \nabla_{\mathbf{x}_k} g(\mathbf{z}_{k+1}^{i,j})\big|_{\mathbf{x}_k^i=\hat{\mathbf{x}}_{k+1|k}^i, \mathbf{x}_k^{i,j}=\hat{\mathbf{x}}_{k+1}^{i,j}} \\ &= \begin{bmatrix} \cos(\hat{\phi}_{k+1}^{i,j}) & -\hat{\rho}_{k+1}^{i,j}\sin(\hat{\phi}_{k+1}^{i,j}) \\ \sin(\hat{\phi}_{k+1}^{i,j}) & \hat{\rho}_{k+1}^{i,j}\cos(\hat{\phi}_{k+1}^{i,j}) \end{bmatrix}. \end{aligned} \tag{7}$$

The jacobian $\mathbf{J}$ relates the deviation of the original $[\Delta\hat{\rho}\ \Delta\hat{\phi}]^\top$ and the transformed $[\Delta\hat{x}\ \Delta\hat{y}]^\top$ variables, which represent the distance from robots i and j in x and y coordinates, respectively. Calculated as:

$$\begin{bmatrix} \Delta\hat{x} \\ \Delta\hat{y} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \Delta\hat{\rho} \\ \Delta\hat{\phi} \end{bmatrix}. \tag{8}$$

The covariance is defined by the expectation of the squared deviates. So, the covariance of the measurement $\mathbf{z}$ in the common frame is defined by multiplying both sides of Eq. (8) by their respective transposes and taking the expectation of the result. This transformation represents an adequate linear approximation when the variables are represented by Gaussians with small variances, as stated in [13]. The uncertainty $\mathbf{P}_{k+1}^{i,j}$ related to the $\hat{\mathbf{x}}_{k+1}^{i,j}$ estimate is generated by the combination of the covariance matrices:

$$\mathbf{P}_{k+1}^{i,j} = \mathbf{P}_{k+1|k}^{i} + \mathbf{J}_{k+1}^{i,j}\mathbf{R}_{k+1}^{i,j}(\mathbf{J}_{k+1}^{i,j})^\top. \tag{9}$$

The estimates $\hat{\mathbf{x}}_{k+1|k}^{i}$ and $\hat{\mathbf{x}}_{k+1}^{i,j}$ are combined by the EKF update step. This generates a new state $\hat{\mathbf{x}}_{k+1|k+1}^{i,j}$ and covariance $\mathbf{P}_{k+1|k+1}^{i,j}$, which represent the actual belief of the robot. New landmark information and relative measurements are combined with this belief using the procedure described above in an incremental way. The final state $\hat{\mathbf{x}}_{k+1|k+1}^{i}$ and covariance $\mathbf{P}_{k+1|k+1}^{i}$ estimate is used for the next motion step.

An important point of the methodology is the choice of the neighbors used as landmarks. We use two different methods: the first one considers the k closest neighbors while the second choses the k neighbors with lowest uncertainties (covariance trace). The performances of these different methods are compared in Sect. 4.

3.3 Swarm Motion Strategy

As mentioned, the swarm $\mathcal{R}$ is randomly divided into two subgroups: one that moves while the other maintains its position. After a pre-specified number of time-steps, the two subgroups exchange roles. Lets call these subgroups $\mathcal{R}_m$ and $\mathcal{R}_s$, for *moving* and *stationary* ones, respectively.

Robots motion is governed by decentralized flocking rules, which allow them to move in a cohesive way, while avoiding collisions. The motion strategy is based on some of the basic rules of Reynolds' flocking algorithm [9]. Each rule establishes a vector that determines a direction to be followed.

The first rule, separation, aims to maintain a safe distance among the robots. The separation behavior is calculated by:

$$\mathbf{v}_i^{sep} = \sum_{j \in \mathcal{R}, j \neq i} \left(\frac{r_s}{\|\mathbf{d}_{ij}\|} - 1 \right) \mathbf{d}_{ij}, \ \|\mathbf{d}_{ij}\| \leq r_s, \tag{10}$$

which computes a separation vector $\mathbf{v}^{sep}$ based on the displacement $\mathbf{d}_{ij}$ between a robot and its teammates located inside a specific separation range (r_s).

The second rule aims to maintain the robots together acting as a unit. This cohesion rule computes the average position of the k moving robots that are located inside a cohesion range (r_c), and generates a vector $\mathbf{v}_i^{coh}$ pointing in this direction as:

$$\mathbf{v}_i^{coh} = \frac{1}{k} \sum_{j \in \mathcal{R}_m, j \neq i} \mathbf{d}_{ij}, \ \|\mathbf{d}_{ij}\| \leq r_c. \tag{11}$$

Finally, we consider that the robots have a series of common targets to be reached during their motion. These targets are used to compute a direction vector ($\mathbf{v}^{dir}$) to be followed. The displacement vector between a robot i and its next target t is represented by $\mathbf{d}_{it}$. As shown in Eq. (12), the direction vector consists of the unit vector in the direction of $\mathbf{d}_{it}$:

$$\mathbf{v}_i^{dir} = \frac{\mathbf{d}_{it}}{\|\mathbf{d}_{it}\|} \ . \tag{12}$$

The control action $\mathbf{u}_i$ is given by:

$$\mathbf{u}_i = k_c \ \mathbf{v}_i^{coh} + k_s \ \mathbf{v}_i^{sep} + k_d \ \mathbf{v}_i^{dir} \ , \tag{13}$$

which is composed by the weighted sum of the three vectors. The control action is then decomposed in velocities vx_i and vy_i, that will be used by robot i.

We assume all robots have a synchronized clock, and the decision to switch robots from $\mathcal{R}_s$ to $\mathcal{R}_m$ (i.e. from stationary landmarks to the moving robots group), and vice versa, is made periodically on a completely decentralized manner. At first, the swarm is divided in two predefined subgroups, and a common timer is initialized.

As soon as the timer reaches a certain value, the robots change their roles, and the timer is reinitialized. This loop is repeated until the mission is fulfilled.

3.4 Complexity Analysis

In this section, we present a brief analysis of the computational complexity of each stage of the proposed methodology, as well as bounds regarding the number of messages used by each robot.

In most systems dealing with the CL problem, the cost to estimate the position is usually $O(n^2)$, where n is the number of robots, since $n(n-1)$ measurements are needed to calculate a new estimation.

In this work, the swarm is divided into two subgroups, and all the robots on a subgroup should measure the relative range and bearing of all robots from the other subgroup. Therefore, it takes $(n/2)(n/2)$ measurements, still leading to an $O(n^2)$ complexity, where n is the number of robots. However, as will be shown in Sect. 4, the use of just part of the robots from the other group improves significantly the position estimation, which takes $k(n/2)$ measurements. Therefore, we have an $O(kn)$ complexity, since we consider $k \ll n/2$ as the number of robots that will be used as landmarks, and n is the total number of robots.

4 Experiments

Several simulations were performed to analyze how the number of dynamic landmarks and the way they are chosen impact the localization. We have also varied parameters related to the quality of the information in order to study its influence on the localization error and the number of necessary landmarks.

The experiments presented here were executed considering a swarm with 30 holonomic robots. The group navigates in an obstacle-free static environment of approximately one hundred square meters (10 m $\times$ 10 m). Robot motion is directed by a series of waypoints, which define targets to be reached by the group. These targets dictates the preferred direction ($\mathbf{v}_i^{dir}$) for each robot and it is used in computing the commanded velocity $\mathbf{u}_i$. These velocities are limited to 0.1 m/s, and are subjected to additive zero-mean Gaussian noise of 10 % of the true velocity. Each simulation takes approximately 6500 time-steps, where $\Delta k = 0.1$ s is the duration of each time-step, and the total length of the traveled distance is $\approx$30 m.

Following the methodology, the robots are initially assembled together and divided into two subgroups at random. The motion of the groups is coordinated so that they move in turns for a specific number of time-steps. One subgroup is selected to start moving while the other remains stationary. After motion, the group stops and each member computes range and bearing to some of the static robots located inside their

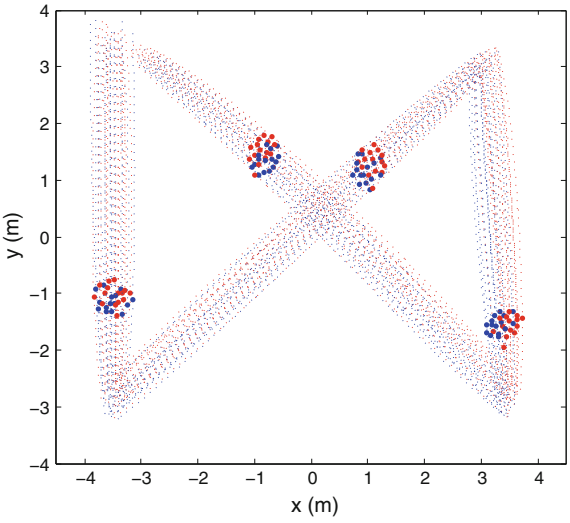


Fig. 1 Example of the path executed by the subgroups (*red* and *blue* colors)

neighborhoods. In the simulations, the radius of the neighborhoods was made large enough to allow the analysis of the impact of varying the number of landmarks.

Figure 1 presents an example of a path executed by the swarm. The position of the members of both groups (depicted in red and blue) is presented over time. The small circles represent the position of each robot in four distinct moments in which the blue group performed cooperative localization.

The first experiment was performed to evaluate the impact of different error parameters in the quality of the final localization. The range noise was defined proportional to the measurements while the bearing noise has been fixed to some absolute values. The noise $\mathbf{n}$ of the measurement model (Eq. 3) was defined using the following values $\sigma_\rho = \{0.05, 0.10, 0.20, 0.30, 0.40\} \times \rho$ and $\sigma_\phi = \{1°, 2°, 3°\}$. These parameters were analyzed varying the number of landmark robots used for correcting the localization estimates, ranging from 1 robot to $\frac{\eta}{2}$ robots (i.e. the entire stationary group). For each specified combination of noise values $(\sigma_\rho, \sigma_\phi)$ and number of landmarks, a total of 30 simulations were executed. The position mean error after the path has been completed is presented in Fig. 2.

For each simulation, a mean localization error is calculated by taking the average of the root mean square error (RMSE) value of the position errors for all robots during their motion, assuming they are using a determined number of neighbors as landmarks. It is possible to observe that the bearing noise has a small influence in the mean error. However, with the increase in the noise of the relative range, a robot needs to use more landmarks in order to better correct its position estimate. We can also observe that in situations where the noise is small, the increase in the number of landmarks over certain values does not contribute to reduce the localization error. For example, for $\sigma_\rho = 0.05\rho$, the error decreases very slowly for more than 6

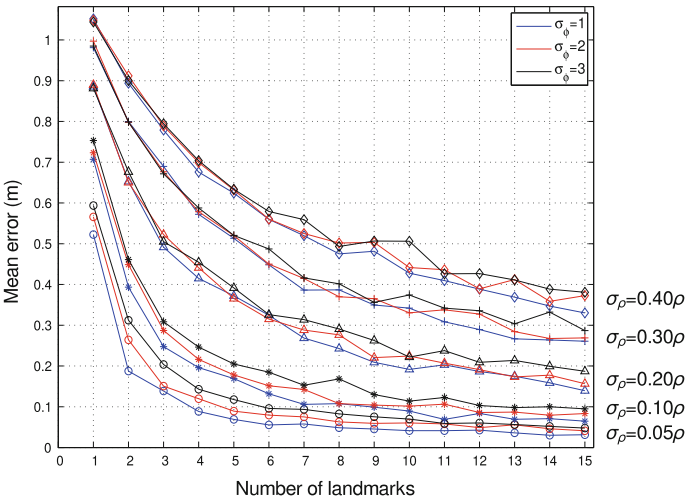


Fig. 2 Mean position error varying the number of landmarks used for correction

landmarks. In other words, depending on the observation noise, robots can improve their localization using just a small number of landmarks in their neighborhood.

We also analyzed the increase on the RMSE along the execution when a different number of landmarks is used. Figure 3 shows the results of a set of 30 simulations in which the associated noises were fixed at $\sigma_\rho = 0.10\rho$ and $\sigma_\phi = 3°$. It is possible to observe that the error is largely reduced when at least one landmark is used, but this reduction is smaller for more than four landmarks. This reinforces the analysis that a small number of robots may suffice for improving localization in swarm navigation.

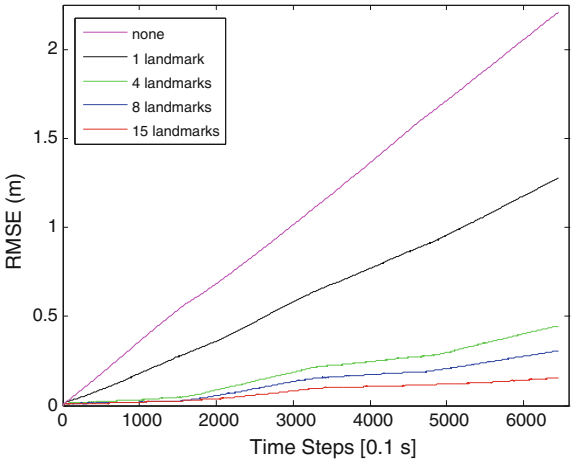


Fig. 3 RMSE accordingly to the different number of landmarks used for localization

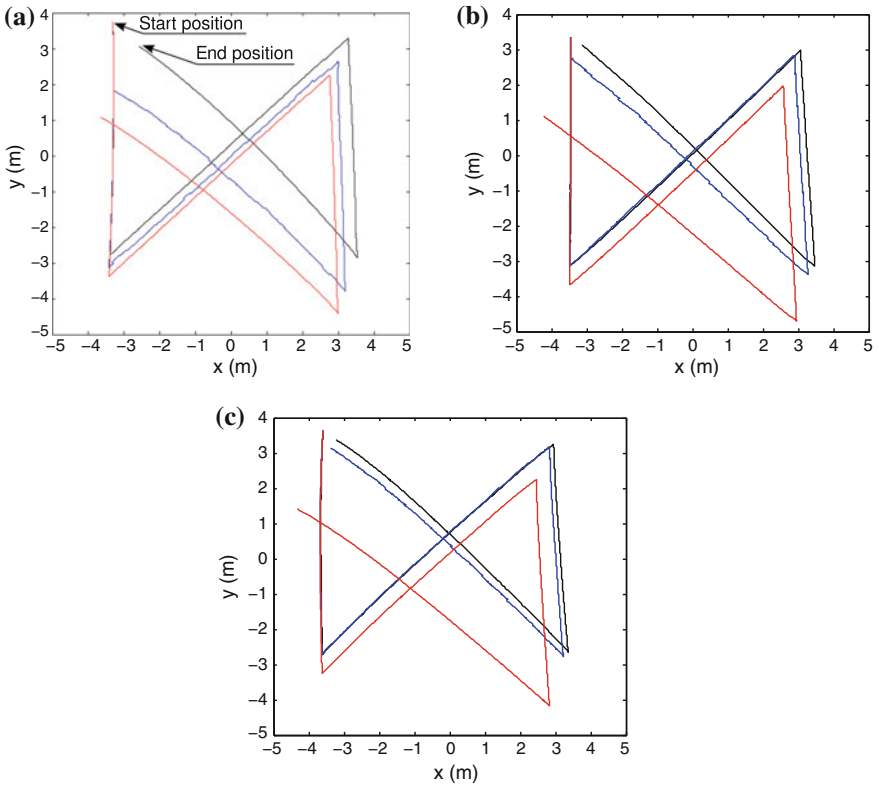


Fig. 4 The real (*black line*) and cooperative estimated (*blue line*) path performed by a robot. The *red line* is the estimate without cooperative localization. **a** 1 landmark. **b** 4 landmarks. **c** 8 landmarks

As can be seen, the RMSE considering the use of 15 landmarks presents a tendency to converge to a constant growing rate after a certain number of time steps. This is a particularly important result, since it provides a notion that it will be possible to obtain a bounded maximum error for the entire swarm.

The improvement on the localization can readily be observed in Fig. 4, which shows the path followed by one of the robots using different numbers of landmarks to correct its localization. In this case, the black line is the ground truth (actual path), the red line represents the robot localization estimates without correction and the blue line represents the localization using other robots as landmarks. Both Figs. 3 and 4 are related to the same set of experiments.

In the previous experiments, the landmarks to be used were chosen at random, without any criterion of selection. The measurements were used in the update phase of the EKF according to the order the messages arrived. With the aim of analyzing how a different ordering could impact the localization, we defined two criteria for selecting the landmarks to be used, and compared them with the random selection. The first considers an ordering process in which the landmarks that are closer to the robot are chosen. The second sorts the landmarks considering their uncertainties, so that

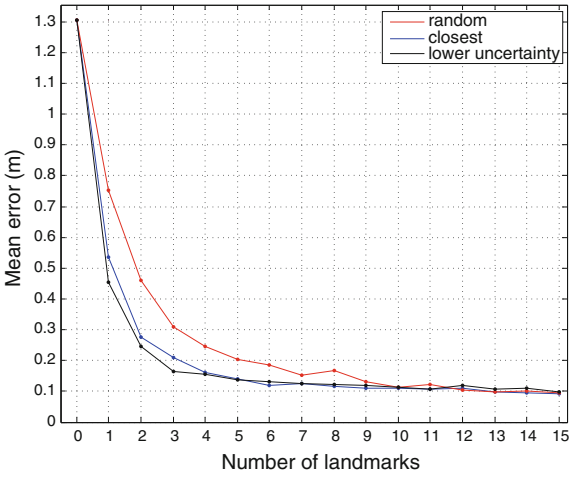


Fig. 5 Mean position error accordingly to the criterion used to select the landmarks

the ones with the lower uncertainty are chosen. The observation noise was set with the same value of the previous experiment. Figure 5 shows that both criteria improve the localization over choosing landmarks randomly. Between the two, the selection of the landmark robots with lower uncertainties is slightly better than choosing the closest landmark robots, specially when few landmarks are used.

5 Coclusion and Future Work

In this paper we performed a characterization of Cooperative Localization for swarms of robots by using an approximate decentralized algorithm. Considering the sensory and computational limitations of this kind of system, we have explored the coordinated motion of the group, so that the robots could cooperatively localize themselves using local information, i.e. using only part of the group as landmarks. We performed a series of simulations in order to evaluate the method and results showed that localization estimates can be significantly improved even using a small number of neighbors as landmarks. Experiments performed with a larger number of robots (not showed here) confirmed this trend. This suggests that the method scales well and can be used when it is necessary to cooperatively localize large groups of robots.

In general, cooperative localization methods have not been used with large group of robots due to the complexity related to the exchange of localization information. Therefore, one of the main contributions of this work is the investigation of such methods in robotic swarms. Although we have not considered the localization interdependence, our results indicate that it is possible to reach good localization even with a few landmarks, which is important in terms of scalability.

The main limitation of this work is related to the fact that we do not incorporate the orientation noise and localization interdependence. In order to address this,

we will direct our future work to extend the proposed methodology to deal with non-holonomic robots and also with the localization interdependence. Specially to tackle the latter, we intend Covariance Intersection Algorithm (CI) [1, 3]. We believe that this will make the methodology more general and robust.

Acknowledgments This work was developed with the support of CEFET-MG, CAPES, FAPEMIG and CNPq.

References

1. Carrillo-Arce, L., Nerurkar, E., Gordillo, J., Roumeliotis, S.: Decentralized multi-robot cooperative localization using covariance intersection. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1412–1417 (2013)
2. Hidaka, Y., Mourikis, A., Roumeliotis, S.: Optimal formations for cooperative localization of mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4126–4131. Barcelona, Spain (2005)
3. Julier, S.J., Uhlmann, J.K.: A non-divergent estimation algorithm in the presence of unknown correlations. In: Proceedings of the American Control Conference, vol. 4, pp. 2369–2373 (1997)
4. Kalman, R.E.: A new approach to linear filtering and prediction problems. Trans. ASME—J. Basic Eng. **82**(Series D), 35–45 (1960)
5. Kurazume, R., Hirose, S.: An experimental study of a cooperative positioning system. Auton. Robots **8**(1), 43–52 (2000)
6. Kurazume, R., Nagata, S., Hirose, S.: Cooperative positioning with multiple robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1250–1257 (1994)
7. Rekleitis, I.M., Dudek, G., Milios, E.E.: On multiagent exploration. In: Proceedings of Vision Interface, pp. 455–461 (1998)
8. Rekleitis, I.M., Dudek, G., Milios, E.E.: Multi-Robot cooperative localization: a study of trade-offs between efficiency and accuracy. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2690–2695 (2002)
9. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model. In: Computer Graphics, pp. 25–34 (1987)
10. Roumeliotis, S.I., Bekey, G.A.: Collective Localization: a distributed Kalman filter approach to localization of groups of mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2958–2965 (2000)
11. Roumeliotis, S.I., Rekleitis, I.M.: Propagation of uncertainty in cooperative multirobot localization: analysis and experimental results. Auton. Robots **17**, 41–54 (2004)
12. Schneider, F.E., Wildermuth, D.: Influences of the robot group size on cooperative multi-robot localisation—analysis and experimental validation. Robot. Auton. Syst. **60**(11), 1421–1428 (2012)
13. Smith, R.C., Cheeseman, P.: On the representation and estimation of spatial uncertainty. Int. J. Robot. Res. **4**, 56–68 (1986)
14. Tully, S., Kantor, G., Choset, H.: Leap-frog path design for multi-robot cooperative localization. In: Howard, A., Iagnemma, K., Kelly, A. (eds.) Field and Service Robotics, Springer Tracts in Advanced Robotics, vol. 62, pp. 307–317. Springer, Berlin (2010)
15. Zhang, F., Grocholsky, B., Kumar, V.: Formations for localization of robot networks. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 3369–3374. New Orleans, LA, USA (2004)
16. Zhou, X.S., Zhou, K.X., Roumeliotis, S.I.: Optimized motion strategies for localization in leader-follower formations. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 98–105. San Francisco, CA, USA (2011)

MOARSLAM: Multiple Operator Augmented RSLAM

John G. Morrison, Dorian Gálvez-López and Gabe Sibley

Abstract To effectively act on the same physical space, robots must first communicate to share and fuse the map of the area in which they operate. For long-term online operation, the merging of maps from heterogeneous devices must be fast and allow for scalable growth in both the number of clients and the size of the map. This paper presents a system which allows multiple clients to share and merge maps built from a state-of-the-art relative SLAM system. Maps can also be augmented with virtual elements that are consistently shared by all the clients. The visual-inertial mapping framework which underlies this system is discussed, along with the server architecture and novel integrated multi-session loop closure system. We show quantitative results of the system. The map fusion benefits are demonstrated with an example augmented reality application.

Keywords Slam · Collaborative SLAM · Multi-agent mapping · Long-term autonomy

1 Introduction

As robotic capabilities increase, cooperative robotic interaction is becoming more attractive. In order to operate in the same environment together, however, robots must simultaneously have an understanding of both the physical environment around them and their location within it. This requires a localization and mapping solution which can be shared between robots and permits relocalization within maps created by other devices. Also, to keep robots affordable, approaches based on low-cost sensors, such

J.G. Morrison (✉) · D. Gálvez-López · G. Sibley
Department of Computer Science, University of Colorado, Boulder, CO 80309, USA
e-mail: john.morrison@colorado.edu

D. Gálvez-López
e-mail: Dorian.GalvezLopez@colorado.edu

G. Sibley
e-mail: gsibley@colorado.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_9

as cameras, are preferable to LIDAR-based solutions because their high cost can be prohibitive when developing distributed robotic applications.

This paper presents a new system, MOARSLAM, for simultaneously building and sharing large-scale 3D maps from multiple devices equipped with either a monocular camera and an IMU or a stereo camera pair. It describes infrastructure, algorithms and core data structures for building and sharing arbitrarily scalable visual maps. The paper also presents a means for distributing physically tethered information, demonstrated here with an augmented reality example.

The presented system operates in a client-server architecture communicating over the network. This work defines a *client* as a camera- and network-equipped platform with computing abilities, such as a mobile phone or an autonomous robot. A *session* represents a single continuous MOARSLAM run by a client. A client starts a new session each time they launch MOARSLAM. The *server* in MOARSLAM is defined as a machine that stores a single representation of the accumulated global map. It offers an API interface to receive and distribute maps to provide endpoints for map related queries. The MOARSLAM client is capable of fully autonomous SLAM and only communicates with the server to share maps. The server's simple and stateless API allows clients to easily integrate server queries into their processing when a network connection is available. The server API is also purely image driven, removing the need for an accurate global location when interacting with the server. Figure 1 shows an example of the capabilities of our approach.

The paper is organized as follows: Sect. 2 presents the related work of multi-device SLAM. Section 3 gives details about the map structure and its benefits. Section 4 gives an overview of the client-side SLAM processing pipeline which generates the initial session's map. Section 5 describes the server component of MOARSLAM which is responsible for storing and joining multiple sessions' maps. Section 6 shows the experimental validation of our proposal, and Sect. 7 includes a discussion on the potential limitations and future work for this approach. Section 8 concludes the paper.

2 Related Work

Multi-device SLAM has been approached in a decentralized fashion by several authors. Several vehicles on a sparse communication network create individual maps that fuse with their neighbors when it is possible, either by following a consensus policy [1, 21], or by distributing data association [3, 12]. Although decentralized systems can reduce the computation effort of each device, the complexity of their communications usually makes it expensive to obtain a global map. On the other hand, Forster et al. [7] and Riazuelo et al. [17] present a multi-threaded method for building maps using Micro Aerial Vehicles (MAVs) and mobile robots with a centralized system. Their approaches both seek to decouple the motion estimation and map building pipelines by communicating keyframe information to a central station where it is fused into a cohesive map structure. Loop closure events and map construction are handled on this central station while the client is left to perform

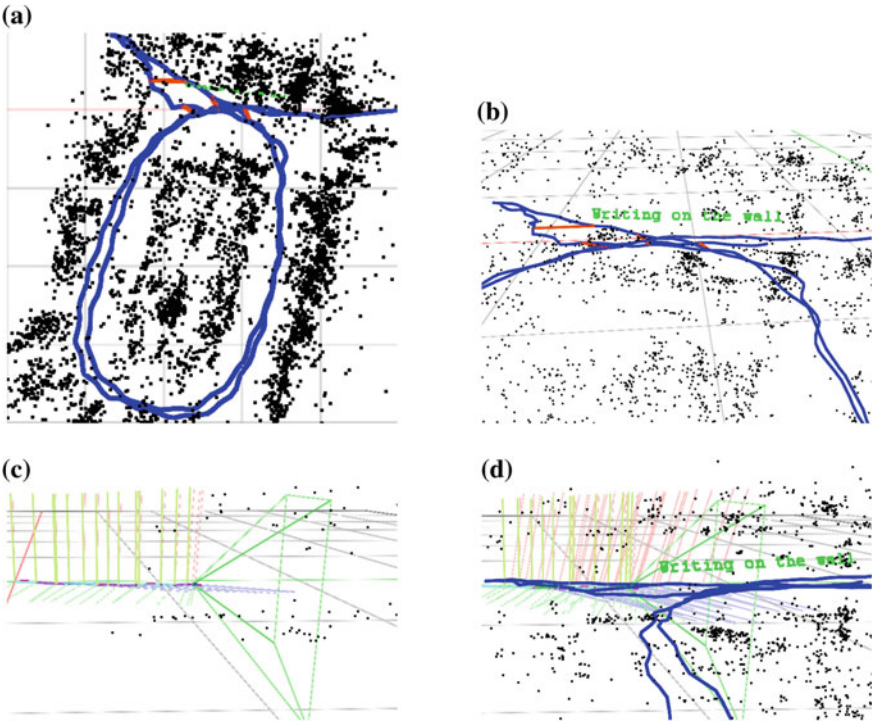


Fig. 1 Sample MOARSLAM operation. **a** A map previously created by a MOARSLAM client is stored in the server. The map and trajectory are depicted as blue edges, and the local loop closures as red edges. The landmarks are shown as black points and a virtual 3D object is also added, displaying the text "Writing on the wall". **b** A zoomed in portion of the map. **c** Another client initiates a new mapping session with MOARLSLAM in the same environment. **d** MOARSLAM soon recognizes a loop closure with the previously mapped trajectory and joins the two maps to provide the client with a richer map

visual odometry with only the map it is provided by the server. These solutions lower the computational cost on the client, but can reduce its autonomy in the face of intermittent networking. The system we propose is also based on a client-server architecture, but unlike approaches above, it allows clients to create individual maps even if connection to the server is lost. When connection is available, individual maps are fused by cross loop closures yielded by the server, as done by McDonald et al. [13]. However, unlike them, we do not require to extract new features to perform this operation since we exploit the features tracked by the front-end. In addition, we reuse the map graph to check for loop consistency.

Previous approaches for map representation fall largely into two camps: "hybrid" sub-mapping techniques and privileged frame fusion approaches. In fully fixed frame representations, such as the one by Özkucur and Akin [16], an expensive map merging algorithm is required to join measurements and objects in multiple maps together.

These approaches offer simplicity in representation, but would not scale well because of the need to reprocess all data before merging maps. The sub-mapping techniques are based on limited-area metric maps, such as the occupancy grids used by Chang et al. [5] or the planar AR workspaces employed in PTAMM [4], the multiple map extension to PTAM [10]. These approaches apply a privileged frame SLAM algorithm to populate their maps, but in order to bound complexity, after a map's extent has grown too large they initialize a new and fully independent metric map. In PTAMM, they reuse their existing relocalization techniques to switch the system's focus between maps, and Chang et al. [5] and McDonald et al. [13] connect maps using topological linkages based on odometry or "anchor nodes". These approaches recognize the bounds of privileged frame representations, but still maintain a dependency on them. In their conclusion Castle et al. [4] discuss future work opportunities that this paper deals with, including IMU integration and the fusion of many sequences.

Current lines of research tend to augment feature maps with additional information, e.g. objects [20]. Furthermore, multiple robot or device localization allows clients to share these data in a consistent manner across maps. For example, the RoboEarth system [22] provides a framework to share semantic maps with object annotations among different robots. However, it does not define a mechanism to build a map cooperatively and requires clients to share an arbitrary reference frame. In contrast, our proposal deals with both cooperative mapping and map reuse, and makes it possible to consistently incorporate meta information relative to each client frame.

3 Scalable Mapping

This section describes the map structure which is the key to enabling multi-device map building, joining and updating in MOARSLAM. First the basic structure of the map is described, followed by a description of the particular attributes of the map structure which make multi-device mapping possible. Finally, the means through which external information is stored in the map is described.

3.1 Map Structure

The map is represented in the "continuous relative representation" of RSLAM [14]: an undirected graph of nodes, representing key frames taken at poses p, connected by transformation edges which encode their 6-DOF relative pose estimate, as seen in Fig. 2. Point landmarks l are stored as inverse-depth estimates back-projected (solid black lines) from the frame where they were first observed [6] back-projected from the pixel where were first seen. Patches of size 9-by-9 pixels are stored for tracking the landmark. Each new observation of a landmark is stored with the frame from

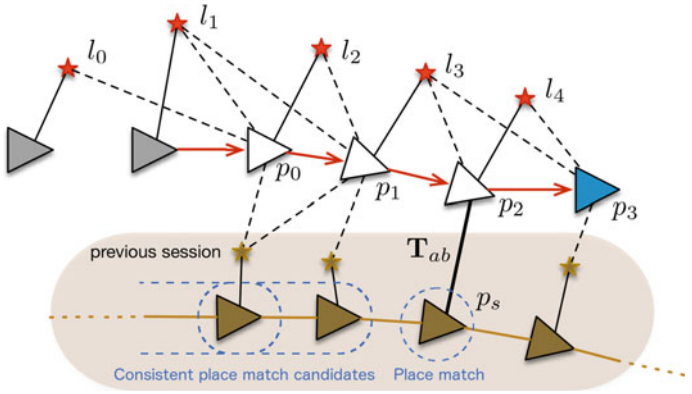


Fig. 2 Graphical representation of a map and of the constraints created after a loop closure. The poses p of a vehicle create graph nodes at every key frame. Nodes are joined by transformation edges and are associated to map landmarks l. When the loop detector retrieves a place match, after several consistent candidate place matches, a new edge T_{ab} is created to fuse maps between the different sessions a and b

which it was observed (dashed edges). Additionally, the camera calibration, both the camera intrinsics and extrinsics, is stored for each session. This allows maps created in separate sessions to be used together. In MOARSLAM, a map is any set of frames which are connected through a set of edges, regardless of the session in which they were created.

3.2 Multi-device Mapping

To enable global uniqueness of information created during each SLAM session, the components of the map require identification. Every time a client begins mapping, it creates a new *Session ID*, a UUID which is used to label every frame, edge, measurement, and landmark created during that session, along with an integer ID making each object unique within a session. The camera calibration used for map creation is also tagged with the session's ID. This identification provides uniqueness of sessions, frames, measurements, and landmark references during processing and transfer so the server and client can be confident that they are referring to the same data.

Using a single relative graph structure to represent a map allows for a consistent interaction between clients, whether information was created locally or fetched from the server. Queries on the map are performed as graph searches. Most searches are in memory and very fast, but the persistent map structure facilitates transparent loading from disk, as described below. This includes searching for visible landmarks, and estimating pose transformations between non-adjacent frames.

Unlike representations based on a single frame of reference, the relative graph structure of MOARSLAM allows sections of mapped environments to be referenced independently of other mapped locations. This means there is no need to share an entire graph or transform nodes before interacting with them. This ability to interact and independently alter sections of the map is important for running many operations in parallel, such as loop closures and multi-session mapping. Also, downloaded sections of map are available for use immediately because the system does not depend on knowledge of a global coordinate frame to make use of frames. Thus, pieces of the global map may be downloaded as they are made available or as bandwidth allows.

As it is designed for large-scale mapping and long-term autonomy, MOARSLAM includes a persistent backend for its map structure. SQLite is used as an interface for persisting portions of the map which are not in use to disk. A simple SQL query of the unique identifiers described above allow frames and edges to be loaded from disk when they are needed. Such a backend structure works well with the sliding window filter optimization in MOARSLAM [14]. Frames are persisted in real time after they have exited the filter and are no longer needed.

3.3 Map Metadata

The relative map framework of MOARSLAM allows physical tagging of 3D space with external information. This information could be anything a robot or an end-user may want to know about a location. This metadata is stored along with a spatial transformation on the keyframe from which its location is visible. In this way, when sharing maps of an environment, all the labeled metadata is distributed with the physical description of the space. In Sect. 6, we demonstrate this capability by simple tagging of locations with augmented reality text, but more complex embedding is possible.

4 Client Processing

MOARSLAM operates in a client-server model. In contrast to Forster et al. [7] and Riazuelo et al. [17] where only a visual odometry system is run on the client, the MOARSLAM client runs a full SLAM pipeline and generates the relative map described in Sect. 3. The overall system architecture is outlined in Fig. 3. The client components are designed with full navigational autonomy in mind. This reduces reliance on potentially intermittent networking capabilities. Each client box in Fig. 3 is divided into several threads, as shown by Table 1.

The client's main responsibility in this framework is to produce relative maps of its environment for sharing. These maps are produced online as the client processes images and IMU data. The client also periodically communicates with the server

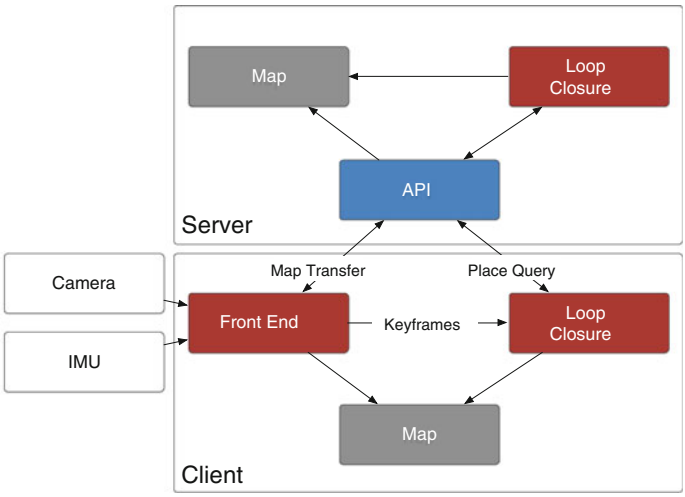


Fig. 3 System architecture outline

Table 1 Threads and algorithms run by a MOARSLAM client

#	Algorithm	Purpose
1	Visual-inertial tracking and estimation [9]	Feature tracking and high-speed SLAM
2	Adaptive-window bundle adjustment [9]	Error reduction in trajectory and landmark estimates
3	Topological loop closure [8]	Previously visited place recognition
4	Post-loop closure map relaxation	Drift reduction through pose graph relaxation
5	Out-of-core SQLite map interface	Reducing memory by writing map data to disk

through the API described in Sect. 5 to query for global loop detections, upload and download maps.

4.1 Client Front-End

The client front-end is responsible for initial processing of all sensory inputs and produces a map. It leverages recent state of the art work on visual-inertial state estimation presented by Keivan et al. [9]. The front-end also asynchronously uploads all recent frames and edges (including their associated metadata) to the server.

The front-end is split into two threads which run asynchronously. The first thread, the tracking thread, tracks image point features using a patch-matching based scheme. This thread is designed to produce approximate pose estimates at a high rate. FAST corner features [18] are extracted from an image and compared with previously found map landmarks. This generates correspondences to estimate an essential matrix

whose decomposition yields a relative pose. We disambiguate the translation scale by incorporating IMU measurements (or by triangulation with a stereo camera). This information is used in a Gauss-Newton optimization step to jointly improve the landmark and pose estimates. Using a set of heuristics based on inter-frame pose differences and the quality of feature tracking, some frames are selected as key frames. For every selected key frame, a local bundle adjustment is run to further improve estimates.

The second thread in the front-end is an asynchronous adaptive-window bundle adjustment thread which grows or shrinks its window to maintain parameter observability and accuracy. This results in very high quality pose estimates at no cost to the overall front-end frame rate. This window can grow from the small size of the synchronous window up to hundreds of key frames, giving it the ability to capture motion and baselines not available to the fast-moving tracking thread.

4.2 Client Loop Closure

A third thread running on the client performs loop closures. For every key frame created during a mapping session, the loop closure system tries to detect places that have already been visited either by the current session or any other. This loop closure thread also queries the server to get matches with other previously uploaded maps. If a match is found, the client will download the server's map of the surrounding area and integrate it into its own, joining the two disconnected maps.

The place matching is fully integrated in the system and takes advantage of all place information available to it, whether it was created during the current session or a previous one. MOARSLAM integrates the loop detector by Gálvez-López and Tardós [8] into its map graph and operates similarly. We store an image database to describe places as bags of binary words by using a single hierarchical vocabulary comprising 10^5 words, trained offline with millions of features obtained from independent data.

The loop closure thread computes ORB [19] descriptors around the corner features tracked by the front-end. This provides a small ($\sim$100) number of points which have been observed from multiple angles, are well distributed across the image and and associated to landmarks with estimated 3D poses. In addition, by reusing the front-end's features the overhead of computing new features is avoided. These descriptors are converted into a bag-of-words vector and used to query the image database. The top-100 candidate matches are grouped together if their reference frames are close in the map, and the group with the highest aggregated similarity score is kept [8]. To avoid false detections, the candidates are accumulated until subsequent queries are resolved, as shown by the blue dashed circles in Fig. 2. When at least 3 of them are close each other, a loop closure attempt is made. By comparing ORB descriptors, 2D-to-3D correspondences between the current image and the matched 3D landmarks are found. Solving the perspective-n-problem yields a relative transformation that is later optimized by projecting map landmarks to find additional correspondences.

If the transformation can be found, the place match is accepted and an edge is added to the map. This is illustrated by the thick edge created between poses p_2 and p_s in Fig. 2 that encodes the spatial transformation T_{ab}. Another thread performs asynchronous map relaxation after a successful loop detection to jointly improve all the pose estimations of the map.

5 Server

The MOARSLAM server component acts as an endpoint for storing, querying, and transmitting the maps which have been built by various MOARSLAM clients. The server provides a stateless API which allows clients to connect and disconnect without affecting the client-server interactions. Through the API, clients can perform three tasks: place recognition queries against previously uploaded maps, map uploading, and map downloading. The server uses the same map structure and place matching approach used on the client, simplifying implementation.

The communication between client and server is implemented using Node [2], an open-source C++ RPC and PubSub framework built on ZeroMQ and Google Protocol Buffers (Figs. 4 and 5).

5.1 API

- **Place Recognition Query**
  The client uploads an image and key frame information to the server to be matched against the server's database of places. The server's place recognition operates as described above in Sect. 4, but manages a larger joint map that comprises the places visited by each client. If a *place recognition query* request is satisfied, an edge is added between two different session maps, as shown in Fig. 2 or as seen in bright green in Fig. 6. A new edge that connects the query frame and the matched frame is returned to the client. Through this edge, the client can perform a *map download* request to access and include landmarks from the downloaded session and use them in its localization.
- **Map Download**
  A client can request a section of graph by specifying a frame ID and a depth to which it would like a breadth-first search to be performed. This search will gather frames, edges, camera calibrations, and all associated metadata, and return it to the requesting client.
  To contain the bandwidth usage of this operation, the maximum size of the returned map is fixed. In those cases that the limit is exceeded, the clients obtain "leaves" of the subgraph, which are the nodes at the edge of the fetched map. This gives the client locations to possibly ask for more map in a new *map download* request, if it is necessary.

- **Map Upload**
  The client serializes frames, edges and associated places and send them to the server for later querying by other devices. These are inserted into the server's databases, which fuses them automatically if there are any conjoining edges.

6 Experimental Evaluation

6.1 Quantitative Results

To demonstrate an aspect of the scalability of MOARSLAM's place query system, this section presents 1099 measurements of the run-time of the a single place query on the server. Each measurement begins after a request to the server has started processing and ends when a match is found or all potential matches have been eliminated. Figure 4a plots the query processing time (in milliseconds) against the number of places in the database. The figure shows the low-constant linear growth rate for the majority of queries. Figure 4b shows the heavy weighting of the measured times towards sub-10 millisecond responses. Both plots also demonstrate the long tail that can occur when interacting with a database, especially a simple on-disk storage interface like SQLite. The MOARSLAM server in this experiment was run on a 4-core 1.60 GHz Core i5 with a 5400 RPM HDD.

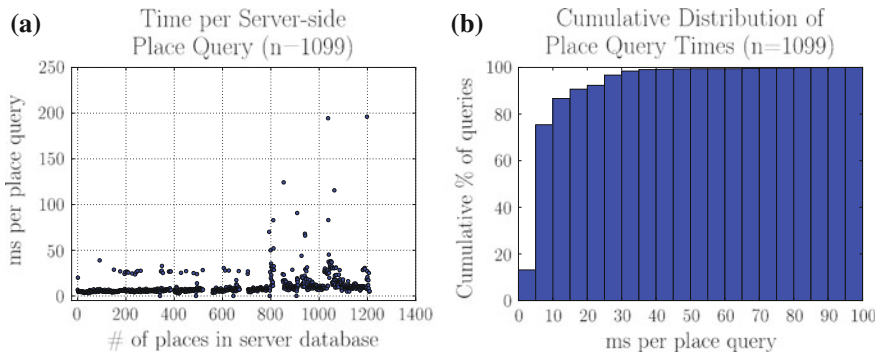


Fig. 4 Plots showing distribution of server-side place recognition query timings. Six data points are not shown on the plot because they fall far outside chart boundaries

Fig. 5 Client 1 created a map with virtual text. Clients 2 and 3 simultaneously downloaded the map, including the virtual text. Clients 2 and 3 are circled in each other's images for perspective. **a** Client 1. **b** Client 2. **c** Client 3

6.2 Qualitative Results

In order to qualitatively validate our multi-device SLAM system, we acquired three sequences of monocular images and inertial data with three Google's Tango mobile phones. These describe different trajectories inside a room of approximately 45 m^2. We then processed these sequences using a local server to create a global map that joins all the data.

We first ran our system with one of the sequences to create an initial map that was uploaded to the server. The two remaining sequences were run as two different client sessions interacting with the server through its API. Figure 6 shows an example of the map fusion process when one of the sessions finds a match with a frame on the server.

To illustrate the ability of our approach to augment maps and provide virtual objects to each client, we inserted a 3D object in the initial environment map displaying the text "Writing on the wall", as it can be seen in Fig. 6. Figure 5 shows a frame of each sequence after fusing maps, showing that the virtual object is correctly located for each client device.

7 Discussion

The relative framework is ideal for scalability and constant-time client operation, but it can present a challenge for users unaccustomed to considering non-euclidean representations. In particular, for *viewing* entire trajectories, it requires a global search and graph-relaxation to construct a consistent global view of the map. As this paper shows, global relaxation is not required for consistent metrically accurate multi-device augmented reality. Further, global relaxation is not needed for path-planning or obstacle avoidance [14]. In fact, a single global coordinate frame makes the estimation brittle, necessitating robust global loop closure algorithms [11, 15] and is only required for visualization. MOARSLAM mitigates the cost of global-optimization

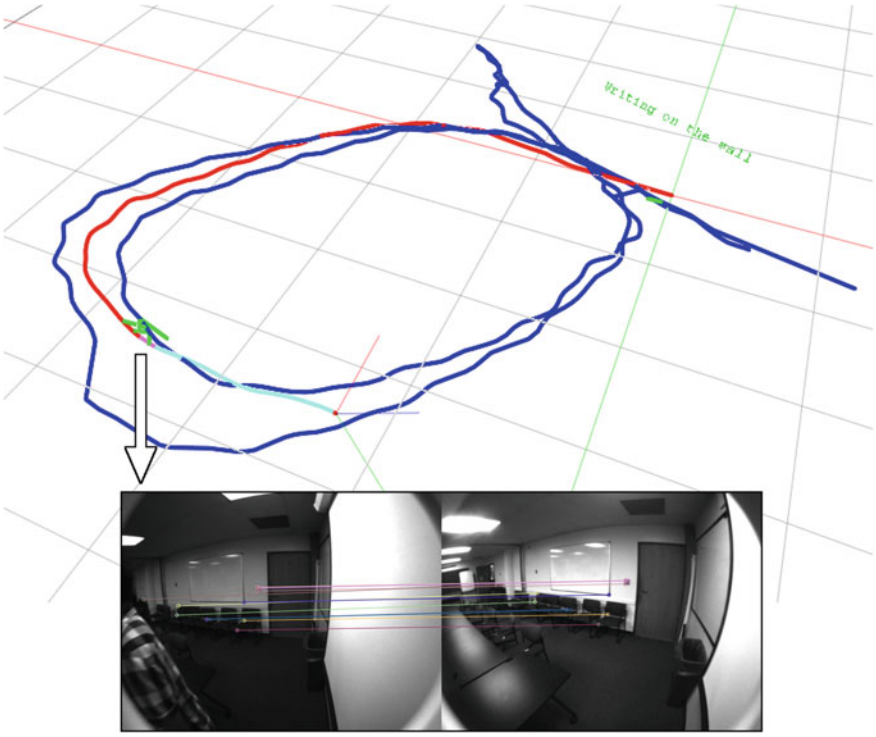


Fig. 6 Two maps are fused after a cross loop detection. The current image of the current session (*red path*) is matched with one the images of a previous session (*blue path*). An edge (*green line*) is added between the matching nodes, encoding a 6-DOF transformation computed from the correspondences between image features and map landmarks

by caching the global map structure and only responding to map updates. It should be emphasized again that viewing global maps is not a requirement for localization or accurate metric interaction with the environment (e.g., for path planning, augmented-reality, obstacle avoidance, etc.). Similarly, while the relative manifold is accurate over short graph traversals of a few kilometers, error can accumulate around loops and cause a "tear" in the global visualization of a map. This is not a problem in the estimation as relative errors are close to optimal, it merely requires global-frame graph-relaxation to produce consistent *visualizations*.

Presently, MOARSLAM's loop detection method is based on a single visual vocabulary. Previous research has shown that this approach is suitable for large heterogeneous environments mapped with very different cameras [8]. The next step is to research the limits of this approach when mapping trajectories of hundreds of kilometers overlapping in urban scenarios. In this context it is important to note that no algorithm is completely exonerated from false positives under all circumstances. Thus, long-term robustness can be achieved by applying a recent technique such as *Realizing, Reversing, and Recovering* (RRR) [11, 15], which accumulates

several loop hypotheses to remove later those that are inconsistent. Note that the relative framework itself is not detrimentally harmed by false loop-closures, which can always be undone, reverting the map to it's prior condition without damaging the state estimate.

8 Conclusion

This paper presented MOARSLAM, a scalable, client-server-based system for multi-device SLAM. In addition, it showed the potential of this system for shared augmented reality (AR) by distributing AR information as a component in the map through experiments demonstrating simultaneous AR from multiple perspectives in an indoor environment.

Sharing SLAM maps in a scalable manner is important for cooperative robotic tasks as robotic platforms become more capable. Sharing physically grounded information will allow teams of robots to operate together in an environment with confidence. MOARSLAM provides a foundation for sharing and reusing the ever more accurate maps created by modern SLAM systems.

Acknowledgments This work is made possible with generous support from Google Project Tango.

References

1. Aragues, R., Cortes, J., Sagues, C.: Distributed consensus on robot networks for dynamically merging feature-based maps. IEEE Trans. Robot. **28**(4), 840–854 (2012)
2. ARPG. Node. https://github.com/arpg/Node
3. Bryson, M., Sukkarieh, S.: Architectures for cooperative airborne simultaneous localisation and mapping. J. Intell. Robot. Syst. **55**(4–5), 267–297 (2009)
4. Castle, R.O., Klein, G., Murray, D.W.: Wide-area augmented reality using camera tracking and mapping in multiple regions. Comput. Vision Image Underst. **115**(6), 854–867 (2011)
5. Chang, H.J., Lee, C.S.G., Hu, Y.C., Yung-Hsiang, Lu.: Multi-robot SLAM with topological/metric maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1467–1472 (2007)
6. Civera, J., Davison, A.J., Montiel, J.M.M.: Inverse depth parametrization for monocular SLAM. IEEE Trans. Robot. **24**(5), 932–945 (2008)
7. Forster, C., Lynen, S., Kneip, L., Scaramuzza, D.: Collaborative monocular SLAM with multiple micro aerial vehicles. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3962–3970. IEEE (2013)
8. Gálvez-López, D., Tardós, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Trans. Robot. **28**(5), 1188–1197 (2012)
9. Keivan, N., Patron-Perez, A., Sibley, G.: Adaptive asynchronous conditioning for visual-inertial SLAM. In: International Symposium on Experimental Robotics, June 2014
10. Klein, G., Murray, D,: Parallel tracking and mapping for small AR workspaces. In: 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234 (2007)
11. Latif, Y., Cadena, C., Neira, J.: Robust loop closing over time for pose graph SLAM. Int. J. Robot. Res. **32**(14), 1611–1626 (2013)

12. Leung, K.Y.K., Barfoot, T.D., Liu, H.H.T.: Distributed and decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks. In: IEEE International Conference onRobotics and Automatio, pp. 3841–3847, May 2011
13. McDonald, J., Kaess, M., Cadena, C., Neira, J., Leonard, J.J.: Real-time 6-DOF multi-session visual SLAM over large-scale environments. Robot. Auton. Syst. **61**(10), 1144–1158 (2013)
14. Mei, C., Sibley, G., Cummins, M., Newman, P., Reid, I.: RSLAM: a system for large-scale mapping in constant-time using stereo. Int. J. Comput. Vision **94**(2), 198–214 (2011)
15. Olson, E., Agarwal, P.: Inference on networks of mixtures for robust robot mapping. Int. J. Robot. Res. **32**(7), 826–840 (2013)
16. Ergin Özkucur, N., Levent Akin, H.: Cooperative multi-robot map merging using fast-SLAM. In: RoboCup 2009: Robot Soccer World Cup XIII, number 5949 in Lecture Notes in Computer Science, pp. 449–460, Jan 2010
17. Riazuelo, L., Civera, J., Montiel, J.M.M.: C2TAM: a cloud framework for cooperative tracking and mapping. Robot. Auton. Syst. **62**(4), 401–413 (2014)
18. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision, pp. 430–443 (2006)
19. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision, pp. 2564–2571. IEEE (2011)
20. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H.J., Davison, A.J.: SLAM++: simultaneous localisation and mapping at the level of objects. In: IEEE Computer Vision and Pattern Recognition, June 2013
21. Sharma, R., Taylor, C., Casbeer, D.W., Beard, R.W.: Distributed cooperative slam using an information consenseus filter. In: AIAA Guidance Navigation and Control Conference, pp. 8334–8342 (2010)
22. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Gálvez-López, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: Roboearth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011)

Part II
Cooperative Manipulation and Task Allocation

Multi-robot Manipulation Without Communication

Zijian Wang and Mac Schwager

Abstract This paper presents a novel multi-robot manipulation algorithm which allows a large number of small robots to move a comparatively large object along a desired trajectory to a goal location. The algorithm does not require an explicit communication network among the robots. Instead, the robots coordinate their actions through sensing the motion of the object itself. It is proven that this implicit information is sufficient to synchronize the forces applied by the robots. A leader robot then steers the forces of the synchronized group to manipulate the object through the desired trajectory to the goal. The paper presents algorithms that are proven to control both translational and rotational motion of the object. Simulations demonstrate the approach for two scenarios with 20 robots transporting a rectangular plank and 1000 robots transporting a piano.

Keywords Multi-robot Manipulation · Cooperative Control

1 Introduction

In this paper we present a scalable, decentralized control strategy by which a large number of robots can manipulate a comparatively large object through a desired trajectory to a goal configuration. The key to the approach is to use the object itself as a medium for transferring information throughout the group of robots. No communication network is required in this strategy. Instead, the robots sense the local motion of the object, and use this information to correct their own force through a feedback law. We prove that this feedback law will cause all robots' forces to align to the same direction exponentially fast. Furthermore, the rate of this exponential convergence increases linearly with the number of robots, so that performance becomes *faster* as the number of robots *increases*, leading to a scalable strategy.

Z. Wang · M. Schwager (✉)
Department of Mechanical Engineering, Boston University, Boston, MA, USA
e-mail: schwager@bu.edu

Z. Wang
e-mail: zjwang@bu.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_10

The forces of the robots synchronize to a leader, which can either be a robot or a human operator. The leader then guides the object through a desired trajectory to the goal configuration, using the synchronized follower robots to multiply its effective force. We provide feedback control laws for the leader to steer the whole system, both in translation and in rotation, under a mild symmetry assumption on the follower robots. We require the leader to know the object's location relative to the desired trajectory, however the follower robots do not need to know their locations, the locations of other robots, the object, or the desired trajectory. We demonstrate the approach in simulations with two scenarios, one involving 20 robots manipulating a plank of wood, and another with 1000 robots manipulating a piano.

Our algorithm is useful in situations where a large number of small, inexpensive robots are required to coordinate to manipulate large objects. For example, in an automated construction site our system could be used to transport massive building materials to a desired location. In a manufacturing facility, this system could be used to transport large products (e.g. aircraft, trains, or industrial equipment) in various stages of assembly to different areas on the manufacturing floor. In our system the leader can be a human operator, enabling one person to move objects that would otherwise require a forklift or a crane. Similarly, in a disaster relief scenario, such a system of robots could be used to autonomously clear debris from a collapsed building to free survivors, or to clean up structurally unstable disaster sites.

The most attractive aspect of our approach is that no explicit communication is needed among robots. Coordinated control algorithms that rely on a wireless network must deal with dropped packets, packet collisions, delays, and fundamental scaling capacity limitations [18]. The presence of a network also requires more sophisticated robots with more sophisticated hardware. In contrast, we take a minimalist approach to achieve scalability. By sensing the motion of the object itself, our follower robots can determine the summed forces of all the other robots acting on the object. This is all the information needed to reach a consensus on the robots' forces. Hence our robots do not require networking hardware, localization information, nor a global reference frame.

1.1 Related Work

Manipulation is a fundamental problem in robotics with an enormous literature, and many algorithms for multi-robot manipulation have been proposed. An early approach to multi-robot manipulation can be found in [15, 16], where various pushing strategies are designed given different availability of sensing and communication. Another approach, known as caging was studied in [1], where a group of robots surround an object and then move together, making sure that the object always remains inside the formation. In this vein, some work has focused on a rigorous geometrical analysis of how the object can be completely caged [2–4]. Although mathematically elegant, these methods typically have considerable computational requirements [5]. On the other hand, some authors have studied formation-based caging, which assumes

that there are enough robots so that the object cannot escape the formation. The object can then be transported by moving the formation as a rigid body using, for example, potential fields [6] or vector fields [7]. Other approaches have used novel modes of actuation, for example tow cables [17, 19]. The approach we describe here is different from all of these in that we require no explicit communication between robots. In this respect, our algorithm is most similar to ensemble control techniques from [8, 9], which also do not require robot-to-robot communication. Our technique is different from ensemble control in that each robot individually steers itself according to its own control actions, as opposed to having one common control signal for the whole group. In addition to the manipulation problem, the similar idea of using local sensor readings as implicit communication was investigated in [23], where an offline evolutionary algorithm was used to design a controller to coordinate the headings of a group of physically connected robots based on inter-robot force measurements.

The analysis of our control strategy takes some inspiration from the study of multi-agent consensus [10, 11] and the study of leader-follower networks [20, 21]. In consensus problems, agents locally exchange information about their neighbors' states through a communication network, in order to reach a consensus on a quantity of interest. Similarly, in leader-follower networks, local communication protocols are used for all agents to converge to the state of a leader. As opposed to these works, we do not have an explicit communication network, however we use analytical tools from this work to show that the robots in our system will converge to the force of the leader robot.

Our work is also inspired by collective ant transport strategies studied in Behavioral Biology and Entomology. Ants, like our robots, have no wireless network, yet they are able to effectively coordinate their actions to manipulate large objects. It was hypothesized in [13] that ants detect small-scale vibration or deformation of the object in order to coordinate their forces. Our algorithm suggests an even simpler hypothesis: ants might synchronize their actions using only the rigid body motion of the object they are trying to transport. In [12] the authors measured the forces exerted by a group of ants and found that ants aligned their forces better and better as the manipulation task went on, which agrees with the synchronization approach we propose here. In addition to translation, [14] determined that only a small number of ants in the group are crucial for the rotation of the object, which is similar to the role of the leader robot in our approach.

The rest of this paper is organized as follows. We model the physics of the object and robots and formally state the problem in Sect. 2. In Sect. 3 we present our multi-robot control strategy for both the followers and the leader and analyze its convergence properties. Finally, in Sect. 4 we show numerical simulations with 20 robots and 1000 robots, and we give our conclusions in Sect. 5.

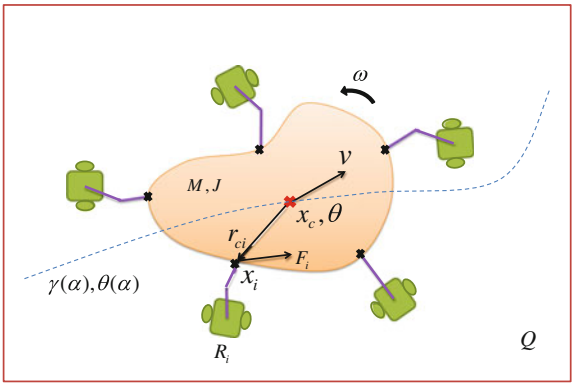


Fig. 1 Configuration of our multi-robot manipulation task. The figure shows an example where five robots (in *green*) are manipulating an object (in *red*)

2 Modeling and Problem Formulation

We consider a planar region $Q \subset \mathbb{R}^2$, where there is a target object with mass M and moment of inertia J, as shown in Fig. 1. The object has three degrees of freedom, that is, the position of the center of mass $x_c \in \mathbb{R}^2$ and the orientation $\theta \in SO(2)$. There is friction force in Q, and we model it as the sum of static friction and viscous friction, whose coefficients are represented as μ_s and μ_v, and the acceleration of gravity is g.

We have a group of N identical robots R_i, $i \in \{1, 2, \ldots, N\}$, trying to transport the object from its initial position to some destination in Q. Each robot is capable of: (i) gripping the object at x_i, which is on the edge of the object, and applying a force $F_i \in \mathbb{R}^2$ in any direction and with any magnitude below its maximum force limit; (ii) measuring the velocity and acceleration, denoted by $v = \dot{x}_c$ and $\dot{v}$, of the manipulated object in the global reference frame[1]; (iii) following the movement of the object such that the desired force can be maintained. We also have a leader robot, indexed as R_1, that is more powerful than the rest of follower robots in that: (i) in addition to applying a force, the leader can also apply a torque $T_1 \in \mathbb{R}$ to the object; (ii) the leader can measure the angle and angular velocity of the object; (iii) the leader knows the desired trajectory, and it can also measure the relative position between the trajectory and the object.

Under the forces from the robots and the environment, the object will move with translational velocity v and angular velocity ω. In the next section, we will derive the translational and rotational dynamics of the object, and then state our problem formally.

[1]This requirement can be relaxed so that robots only know the velocity and acceleration of their attachment point, $\dot{x}_i$ and $\ddot{x}_i$, in their local reference frame, however this considerably complicates the dynamics. We treat the simpler case here for clarity.

2.1 Translational Dynamics

For translation, the forces are either friction or from robots. So the translational dynamics can be written according to Newton's second law

$$M\dot{v} = \sum_{i=1}^{N} F_i - \mu_s M g \frac{v}{\|v\|} - \mu_v v. \tag{1}$$

2.2 Rotational Dynamics

Since the object has geometric extension around the center of mass rather than a point mass, the force applied to the object will generate a torque if it does not point to or from the center of mass. In order to characterize the rotational dynamics, we need to study two types of torques, associated with friction and robots' forces respectively.

Frictional Torque. Here we derive a model of the torque due to viscous friction on the object. We neglect any torque due to static friction, as such effects are difficult to model, and are expected to be small in comparison to viscous and inertial forces. Consider the velocity of an arbitrary point x on the object, which is translating while rotating as shown in Fig. 2,

$$v_a = v + \omega \times r,$$

where v_a is the absolute velocity, v is the translational velocity at x_c, r is the vector pointing from x_c to our selected point. Note that v_a is different for different points on the object. Then the viscous friction at any selected point can be written as

$$F_v = -\mu_v v_a = -\mu_v v - \mu_v(\omega \times r). \tag{2}$$

The total frictional torque can then be calculated by taking the integral of (2) over the object's bottom surface. Our conclusion is that the torque generated by the viscous friction is proportional to the angular velocity, as shown below.

Proposition 1 (Frictional Torque) *Given an object with arbitrary shape in Q, denote the torque caused by the viscous friction by T_f. Then*

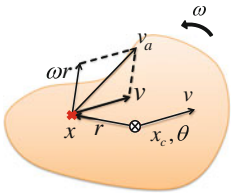


Fig. 2 The synthesis of the absolute velocity of an arbitrary point on the object

$$T_f = -\frac{\mu_v}{M} J\omega. \tag{3}$$

Proof In (2), the viscous friction F_v has two parts, $-\mu_v v$ and $-\mu_v(\omega \times r)$. Denote the torques associated with them by T_{f1} and T_{f2} respectively. Denote density of the object by ρ, so we know the center of mass x_c can be represented as

$$x_c = \frac{\int_S \rho x dx}{\int_S \rho dx} = \frac{\int_S \rho x dx}{M}.$$

Then we have

$$\begin{aligned} T_{f1} &= -\int_S \frac{\rho\mu_v}{M}(r \times v_e)dr = -\int_S \frac{\rho\mu_v}{M}\left[(x - x_c) \times v_e\right]dx \\ &= -\frac{\mu_v}{M}\int_S \rho x \times v_e dx + \frac{\mu_v}{M} x_c \times v_e \int_S \rho dx = -\frac{\mu_v}{M}\left(\int_S \rho x dx\right) \times v_e + \mu_v x_c \times v_e \\ &= -\frac{\mu_v}{M} M x_c \times v_e + \mu_v x_c \times v_e = 0. \end{aligned}$$

As for T_{f2}, note that in the object's local reference frame, ω is always perpendicular to r, so we have

$$T_{f2} = -\int_S \frac{\rho\mu_v}{M}(\omega \times r) r dr = -\int_S \frac{\rho\mu_v}{M}\omega r^2 dr = -\left(\frac{\mu_v}{M}\int_S \rho r^2 dr\right)\omega = -\frac{\mu_v}{M} J\omega.$$

Finally we have $T_f = T_{f1} + T_{f2} = T_{f2} = -\frac{\mu_v}{M} J\omega$. □

Robots' Torques. Torques from robots also have two parts. The first part comes from the forces applied by every robot on the edges of the object. The second part comes from the leader robot's direct torque input. The robots cannot compute the first part because we assume that they do not know their position on the object, r_{ci}. However, with many robots equally spaced around the perimeter, these torques would cancel out exponentially fast. This is formalized in the following symmetry assumption.

Assumption 1 (*Centrosymmetric*) The distribution of the positions of robots' forces is centrosymmetric around x_c. In other words, $\sum_{i=1}^{N} r_{ci} = 0$.

Under Assumption 1, when consensus is reached, all F_i will be the same and thus the resulting torque is zero, which is shown in Fig. 3. In contrast, in Fig. 4 where the forces are not centrosymmetric, the torque is not necessarily zero. In realistic situations, if the number of robots is large with respect to the size of the object, then it is more likely that Assumption 1 will be satisfied or nearly satisfied. However, Assumption 1 does not guarantee that the torque caused by all F_i will always be zero. For example, when the leader's force changes dramatically in a turning process, it will take some time for followers to track the leader and reach the consensus although this process is exponentially fast. We view it as the modeling error, which can be dealt with by the robustness of our controller.

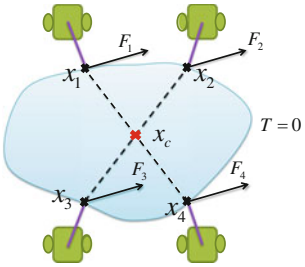


Fig. 3 Centrosymmetric forces

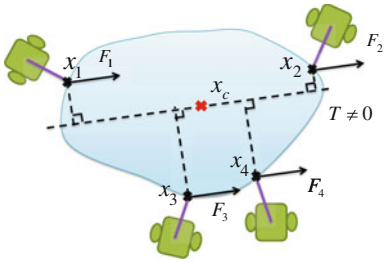


Fig. 4 Non-centrosymmetric forces

Therefore, the only torque we consider from robots is the leader's direct torque input T_1. Hence, the overall rotational dynamics can be written as

$$J\dot{\omega} = T_1 - T_f = T_1 - \frac{\mu_v}{M} J\omega. \tag{4}$$

2.3 Problem Formulation

We assume that the object is too heavy for any individual robot to move due to the static friction force. However, if all the forces from every robot are aligned, the sum of these forces can overcome the static friction, i.e.,

$$max\{\|F_i\|\} < \mu_s Mg \tag{5}$$

$$\frac{\mu_s Mg}{max\{\|F_i\|\}} < N. \tag{6}$$

The goal is to coordinate the forces from all robots to transport the object along a desired trajectory. We define the desired trajectory as

$$\begin{aligned} \gamma(\alpha) : [0, 1] &\mapsto Q \\ \theta &= g_\theta(\gamma), \end{aligned} \tag{7}$$

where α is the index of the point on the trajectory while g_θ is the function that specifies the desired angles for different points on the trajectory. For example, $\gamma(\alpha = 0)$ is the start of the trajectory, and $\gamma(\alpha = 1)$ is the end of the trajectory. Note that the desired trajectory is given by other path planners, such as [22].

Robots only know parameters M, μ_s, μ_v, g, N, but they do not know the global position of the object, their own, other robots, nor where they attach to the object, i.e., x_i. There is no explicit communication between any two robots and the desired trajectory is only known to the leader robot.

3 Control Strategy and Analysis

In this section, we will present how to coordinate the forces from all robots using consensus but without communication. Using consensus, we can transform the original $2N$-dimensional force into a reduced state representation. Furthermore, we treat the leader robot's force and torque as the inputs of the entire system while the linear and angular velocity of the object are the outputs. The overall dynamics is studied and enables us to design the controller for trajectory following based on it.

3.1 Force Coordination with Consensus

Consensus in our case means that all robots will eventually apply the same forces to the object. In conventional consensus methods, explicit communication is required among agents to exchange state information. In contrast, we avoid explicit communication by using the local measurement of the object's movement.

At the beginning of the task, the motion of the object can be initiated randomly.[2] Once the object starts to move, all the robots use the following force updating law

$$\begin{aligned}\dot{F}_i(t) &= \sum_{j=1, j\neq i}^{N} \big(F_j(t) - F_i(t)\big) \\ &= \sum_{j=1}^{N} F_j(t) - N F_i(t) = M\dot{v} + \mu_s M g \frac{v}{\|v\|} + \mu_v v - N F_i(t).\end{aligned} \tag{8}$$

Eq. (8) is computable by every robot since all the terms are locally known without communication. Note that in practice, the robots' forces are not directly addable since they are in different local reference frames. However, for the convenience of

[2]For example, all the robots can repeatedly apply forces in random directions. Eventually enough of the forces will algin by chance to overcome static friction, and the object will begin to move.

analysis, in (8) we express the forces in the global reference frame, which does not effect the correctness of the analysis.

The first row in (8) is the commonly used consensus protocol. If we stack all the forces into one vector $F(t) = (F_1(t)\ F_2(t)\ \cdots\ F_N(t))^T$, then (8) can be also put into the matrix form

$$\dot{F}(t) = -LF(t) \tag{9}$$

$$L = \begin{pmatrix} N-1 & -1 & \cdots & -1 \\ -1 & N-1 & \cdots & -1 \\ \vdots & \vdots & \ddots & -1 \\ -1 & -1 & \cdots & N-1 \end{pmatrix}, \tag{10}$$

where we can find out that L is the graph Laplacian of a completely connected graph. Since $-L$ is negative semi-definite and (9) is a stable linear system, we have that $F(t)$ will converge to the null space of L, which is spanned by $\mathbf{1}$. More specifically, $F(t)$ will converge to $Ave\big(F(0)\big)\mathbf{1} = \big(\sum_{i=1}^{N} F_i(0)/N\big)\mathbf{1}$, as proven in [10]. This is how we produce the consensus without communication.

The force updating law (8) ensures the force consensus ending up with the average value of the initial forces. However, we also want to steer the consensus in order to get any desired manipulation force for trajectory following. In order to do this, we let the leader robot R_1 make its own decision about what force to apply. If the leader does not change its value, then all the followers will converge to the leader's value, i.e.,

$$\lim_{t\to\infty} F(t) = F_1(0)\mathbf{1}, \tag{11}$$

where $F_1(0) \in \mathbb{R}^2$ stands for the leader's force, as proven in [11]. More generally, if the leader keeps changing its value, the followers will still follow the leader, and we will discuss this in detail in the next section.

3.2 Reduced State Representation

Here we present a reduced state representation in the following theorem. Having the reduced state representation, the changing leader can know how the followers will follow it and what the group force will be, which we will use to control the object's movement.

Theorem 1 (Reduced State Representation) *Given a multi-robot system containing N robots, let robot R_1 be the only leader in the group, and the rest are followers which update their forces using (8). Then the reduced state representation can be written as:*

$$\begin{aligned} \dot{\eta}(t) &= -\eta(t) + F_1(t) \\ F_s(t) &= (N-1)\eta(t) + F_1(t) \end{aligned} \tag{12}$$

where $F_s(t) = \sum_{i=1}^{N} F_i(t)$ *is the group force, and* $\eta(t) = (\sum_{i=2}^{N} F_i(t))/(N-1)$ *denotes the average force of all followers.*

Proof By adding up (8) when i goes from 2 to N we have

$$\sum_{i=2}^{N} \dot{F}_i(t) = (N-1)\sum_{j=1}^{N} F_j(t) - N\sum_{i=2}^{N} F_i(t) = (N-1)\sum_{j=1}^{N} F_j(t) - N(\sum_{j=1}^{N} F_j(t) - F_1(t))$$
$$= -\sum_{j=1}^{N} F_j(t) + NF_1(t) = -\sum_{j=2}^{N} F_j(t) + (N-1)F_1(t).$$

Hence we have $\dot{\eta}(t) = -\eta(t) + F_1(t)$, and $F_s(t) = (N-1)\eta(t) + F_1(t)$ since $F_s(t)$ is the sum of followers' forces and the leader's force. □

We can see that by choosing followers' average force as the state variable, the group force can be put in the format of a standard linear control system $\dot{x} = Ax + Bu,\ y = Cx + Du$. As such, the dynamics from the leader's input force to the resulting group force is first-order, meaning that the leader robot can easily implement feedback control to steer the group force with desired specifications.

Furthermore, the internal state η can act as a good approximation of the force of any individual follower robot, as illustrated by the following theorem.

Theorem 2 *The difference among all followers in (9) will converge exponentially to zero regardless of the leader's input. The rate of this exponential convergence increases linearly with the number of robots.*

Proof Consider any two follower robots R_i and R_k, according to (8)

$$\dot{F}_i(t) - \dot{F}_k(t) = -N(F_i - F_k).$$

So we have

$$F_i(t) - F_k(t) = Ce^{-Nt}.$$

□

Theorem 2 implies that after a quick transience, all the followers' forces will be the same, so that η will be approximately equal to any follower robot's force. Continuing on Theorem 2, we can show that the convergence of the followers' forces to the leader's force is also faster as the number of robots increases. Taking the derivative of F_s in (12) we have $\dot{F}_s = (N-1)\dot{\eta} + \dot{F}_1 = (N-1)(F_1 - \eta) + \dot{F}_1$, where $(F_1 - \eta)$ is the difference between the leader and followers and $(N-1)$ works as the feedback amplifying coefficient for the difference. Therefore the larger N is, the faster F_s will be driven to the desired value.

3.3 Controller Design and Trajectory Following

Putting everything together, we can write down the overall state-space dynamics of the system, and derive a controller based on it. The inputs of the system are the leader robot's force F_1 and torque T_1. The outputs are the object's linear and angular velocity, v and ω.

Choose v, ω, η as state variables and combine (1), (4) and (12), we get

$$\begin{pmatrix} \dot{\eta} \\ \dot{v} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ \frac{N-1}{M} & -\frac{\mu_v}{M} & 0 \\ 0 & 0 & -\frac{\mu_v}{M} \end{pmatrix} \begin{pmatrix} \eta \\ v \\ \omega \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ \frac{1}{M} & 0 \\ 0 & \frac{1}{J} \end{pmatrix} \begin{pmatrix} F_1 \\ T_1 \end{pmatrix} + \begin{pmatrix} 0 \\ -\mu_s g \frac{v}{\|v\|} \\ 0 \end{pmatrix}. \quad (13)$$

There are a few remarks on the dynamics above: (i) the rotation and translation dynamics are independent from each other, although we write them together for simplicity and clearity; (ii) the nonlinear term induced by the static friction, $-\mu_s g v/\|v\|$, can be compensated by offsetting F_1, such that the overall dynamics is still linear. Here we briefly show how the compensation works. Let the compensated force $F_1' = F_1 + \mu_s g v/(N\|v\|)$ and divide η into two parts: $\eta = \eta_1 + \eta_2$, where $\dot{\eta}_1 = -\eta_1 + F_1$, $\dot{\eta}_2 = -\eta_2 + \mu_s g v/(N\|v\|)$. Note that η_2 is not affected by F_1 so we have $\eta_2 \to \mu_s g v/(N\|v\|)$. Therefore according to (13), $\dot{v} = \frac{N-1}{M}(\eta_1 + \frac{\mu_s g v}{N\|v\|}) - \frac{\mu_v}{M} v + \frac{1}{M}(F_1 + \frac{\mu_s g v}{N\|v\|}) = \frac{N-1}{M}\eta_1 - \frac{\mu_v}{M} v + \frac{1}{M} F_1$, and we can see that the nonlinear term is eliminated.

We use state feedback to achieve the desired system performance. Let $F_1 = K_f(v_d - v) + \mu_s g v/(N\|v\|)$ and $T_1 = K_t(\omega_d - \omega)$, where v_d and ω_d come from higher-level path planning algorithm. Then the objective is to calculate K_f and K_t according to our specifications. Note that the state feedback only involves proportional control. Integral and derivative control can be implemented by introducing new state variables that are the integral or derivative of the error signal.

Having the controller for v and ω, we can now move on to trajectory following. This requires specifying the desired values of v_d and ω_d. For v_d, we use a straightforward vector synthesis strategy. Firstly, we define the point on the desired trajectory that is nearest to object's current position:

$$x_a = \underset{\gamma(\alpha), \alpha \in [0,1]}{\operatorname{argmin}} \|\gamma(\alpha) - x_c\|.$$

Then the desired velocity can be defined as $v_d = w_n v_n + w_t v_t$, as shown in Fig. 5, where $v_n = \frac{x_a - x_c}{\|x_a - x_c\|}$, v_t is the unit tangential vector at x_a pointing to the destination and w_n, w_t are just weights. Intuitively, v_n will drag the object towards the trajectory and v_t will maintain the object's velocity along the trajectory. As for rotation, we define $\omega_d = k_\theta(\theta_d - \theta) = k_\theta(g_\theta(x_a) - \theta)$, where k_θ is a constant gain.

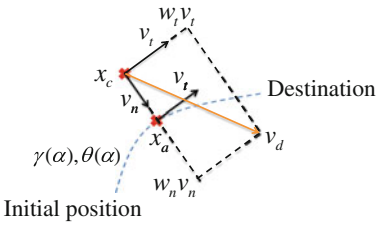


Fig. 5 The synthesis of the desired linear velocity v_d

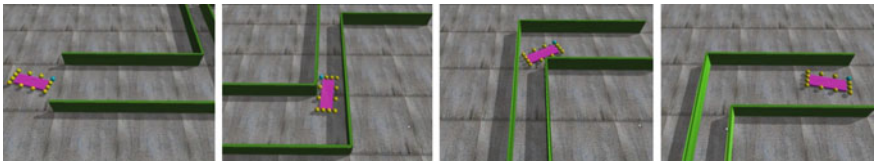


Fig. 6 Simulation 1: manipulation of a small plank (purple) with 12 robots. Dimensions of the object are: weight 1 kg, length 0.6m, width 0.2 m, height 0.1 m. The sphere in *blue* denotes the leader robot while the follower robots are *yellow* spheres. The width of the maze varies from 0.5 to 1 m

4 Simulations

We conduct two manipulation tasks in simulation using Open Dynamic Engine (ODE), a well-known open-source physics engine. The objective of the tasks is to transport an object through an S-shaped maze. In simulation 1, we perform the manipulation for a rectangular plank with 12 robots. In simulation 2, we use 1000 robots to move a large piano of realistic dimensions, which verifies the scalability of our approach. Note that although our controller is derived based on the simplified rotational dynamics (4), the simulator does account for the complete dynamics for the object without any simplification. This suggests that the modeling error of our approach is acceptable and will not significantly affect the overall performance.

The snapshots of simulation 1 and 2 are shown in Figs. 6 and 7.[3] Both the desired and actual trajectories are shown in Fig. 8. The parameters of the environment are: $\mu_s = 0.5$, $\mu_v = 0.3$, $g = 10$. The initial forces of the robots are randomized in the first quadrant, i.e., the angles of the initial forces are in $[0, \frac{\pi}{2}]$. In simulation 1, the force limit of each robot is up to $1.4N$, the torque limit for the leader robot is 5 Nm. In simulation 2, the force limit of each robot is up to $2N$, and the torque limit for the leader robot is 50 Nm. In both simulations, the robotic team successfully transports the object through the maze with rotation being controlled to avoid collision with the wall. Although at the beginning there is a large deviation from the desired trajectory due to the random initial motion of the object, the robots can quickly correct the

[3]The video is available online, http://youtu.be/emZVxcl3Zg4.

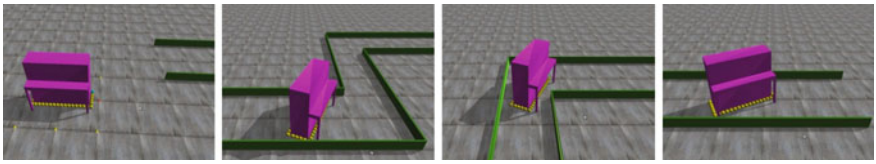


Fig. 7 Simulation 2: manipulation of a large piano (*purple*) with 1000 robots. The dimensions of the simulated piano are the same as a realistic Steinway K-52 piano: weight 273 kg, length 1.54 m, width 0.67 m, height 1.32 m. Robots are centrosymmetrically distributed around the *bottom* of the piano. For visualization considerations, we draw 40 robots instead of 1000. The width of the maze varies from 1.4 to 2 m

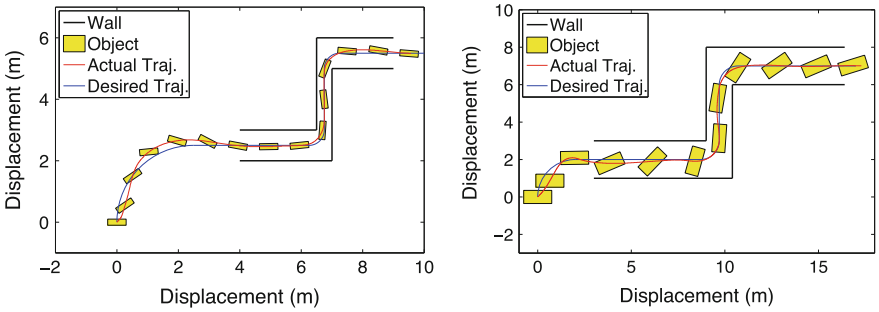


Fig. 8 Overall trajectory of the rectangular plank (*left*) and the realistic piano (*right*)

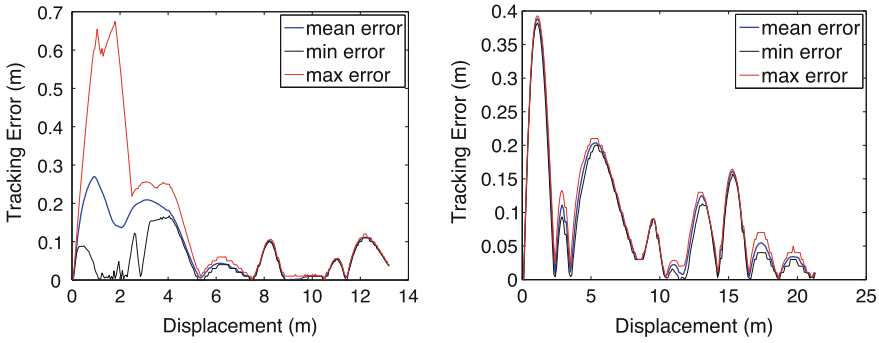


Fig. 9 Trajectory tracking error of consecutive 20 runs for the rectangular plank (*left*) and the realistic piano (*right*)

deviation. This is also revealed in Fig. 9. Notice that the tracking error goes down when on a straight line and goes up when making a turn. Moreover, the variance of the error of the trajectory following is smaller in 1000 robots case than that in 12 robots case, which verifies that the performance of our algorithm improves as the number of robots increases.

5 Conclusion

In this paper, we propose a multi-robot manipulation approach for many small robots to manipulate a massive object. The robots do not have an explicit communication network, but they can locally sense the movement of the object, which gives an indication of the summed forces applied by other robots. We propose a controller by which the robots use the motion of the object itself to reach a consensus on their applied forces. We then design a controller for the leader robot to steer the object based on the analysis of the translational and rotational dynamics of the system. We demonstrate the effectiveness of the approach with two simulations implemented in ODE. In the future, we intend to implement our approach experimentally on mobile robot platforms. We are also investigating using parameter adaptation so that robots do not need to know the mass of the object or friction coefficients beforehand, but can learn these quantities on-line.

Acknowledgments This work was supported in part by NSF grant CNS-1330036. We are grateful for this support. We also would like to thank James McLurkin and Golnaz Habibi for many insightful discussions on this topic.

References

1. Spletzer, J., Das, A., Fierro, R., Taylor, C., Kumar, V., Ostrowski, J.: Cooperative localization and control for multi-robot manipulation. In: Intelligent Robots and Systems, IEEE/RSJ International Conference on (2001)
2. Wang, Z., Kumar, V.: Object closure and manipulation by multiple cooperating mobile robots. In: IEEE International Conference on Robotics and Automation (2002)
3. Pereira, G.A., Campos, M.F., Kumar, V.: Decentralized algorithms for multi-robot manipulation via caging. Int. J. Robot. Res. **23**(7–8), 783–795 (2004)
4. Wan, W., Fukui, R., Shimosaka, M., Sato, T., Kuniyoshi, Y.: Cooperative manipulation with least number of robots via robust caging. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (2012)
5. Wang, Z., Kumar, V., Hirata, Y., Kosuge, K.: A strategy and a fast testing algorithm for object caging by multiple cooperative robots. In: IEEE International Conference on Robotics and Automation (2003)
6. Song, P., Kumar, V.: A potential field based approach to multi-robot manipulation. In: IEEE International Conference on Robotics and Automation (2002)
7. Fink, J., Michael, N., Kumar, V.: Composition of vector fields for multi-robot manipulation via caging. In: Robotics Science and Systems (2007)
8. Becker, A., Habibi, G., Werfel, J., Rubenstein, M., McLurkin, J.: Massive uniform manipulation: controlling large populations of simple robots with a common input signal. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 520–527 (2013)
9. Rubenstein, M., Cabrera, A., Werfel, J., Habibi, G., McLurkin, J., Nagpal, R.: Collective transport of complex objects by simple robots: theory and experiments. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems, pp. 47–54 (2013)
10. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control **49**(9), 1520–1533 (2004)

11. Jadbabaie, A., Lin, J., Morse, A.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Autom. Control **48**(6), 988–1001 (2003)
12. Berman, S., Lindsey, Q., Sakar, M.S., Kumar, V., Pratt, S.: Study of group food retrieval by ants as a model for multi-robot collective transport strategies. Robot. Sci. Syst. (2010)
13. McCreery, H.F., Breed, M.D.: Cooperative transport in ants: a review of proximate mechanisms. Insectes Sociaux (2014)
14. Czaczkes, T.J., Ratnieks, F.L.W.: Simple rules result in the adaptive turning of food items to reduce drag during cooperative food transport in the ant Pheidole oxyops. Insectes Sociaux (2011)
15. Rus, D., Donald, B., Jennings, J.: Moving furniture with teams of autonomous robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (1995)
16. Böhringer, K., Brown, R., Donald, B., Jennings, J., Rus, D.: Distributed Robotic Manipulation: Experiments in Minimalism Experimental Robotics IV. Springer, NewYork (1997)
17. B. Donald, L. Gariepy and D. Rus. Distributed manipulation of multiple objects using ropes. In: IEEE International Conference on Robotics and Automation (2000)
18. Gupta, P., Kumar, P.R.: The capacity of wireless networks. IEEE Int. Conf. Inform. Theory **46**(2), 388–404 (2000)
19. Fink, J., Michael, N., Kim, S., Kumar, V.: Planning and control for cooperative manipulation and transportation with aerial robots. Int. J. Robot. Res. **30**(3), 324–334 (2011)
20. Ji, M., Muhammad, A., Egerstedt, M.: Leader-based multi-agent coordination: controllability and optimal control. In: American Control Conference, pp. 1358–1363 (2006)
21. Tanner, H.G., Pappas, G.J., Kumar, V.: Leader-to-Formation stability. IEEE Trans. Robot. Autom. **20**(3), 443–455 (2004)
22. Habibi, G., Schmidt, L., Jellins, M., McLurkin, J.: K-redundant trees for safe multi-robot recovery in complex environments. In: International Symposium on Robotics Research (2013)
23. Baldassarre, Gianluca, Trianni, Vito, Bonani, Michael, Mondada, Francesco, Dorigo, Marco, Nolfi, Stefano: Self-organized coordinated motion in groups of physically connected robots. IEEE Trans. Syst. Man Cybern. **37**(1), 224–239 (2007)

Distributed Path Planning for Collective Transport Using Homogeneous Multi-robot Systems

Golnaz Habibi, William Xie, Mathew Jellins and James McLurkin

Abstract We present a scalable distributed path planning algorithm for transporting a large object through an unknown environment using a group of homogeneous robots. The robots are randomly scattered across the terrain and collectively sample the obstacles in the environment in a distributed fashion. Given this sampling and the dimensions of the bounding box of the object, the robots construct a distributed configuration space. We then use a variant of the distributed Bellman-Ford algorithm to construct a shortest-path tree using a custom cost function from the goal location to all other connected robots. The cost function encompasses the work required to rotate and translate the object in addition to an extra control penalty to navigate close to obstacles. Our approach sets up a framework that allows the user to balance the trade-off between the safety of the path and the mechanical work required to move the object. The path is optimal given the sampling of the robots and user input parameters. We implemented our algorithm in both simulated and real-world environments. Our approach is robust to the size and shape of the object and adapts to dynamic environments.

Keywords Path planning · Distributed algorithm · Distributed bellman-ford algorithm · Multi-robot system · Collective transport

G. Habibi (✉) · J. McLurkin
Department of Computer Science, Rice University, Houston, TX, USA
e-mail: golnaz.habibi@rice.edu

J. McLurkin
e-mail: jmclurkin@rice.edu

W. Xie
Department of Computer Science, University of Texas at Austin, Austin, TX, USA
e-mail: wxie@cs.utexas.edu

M. Jellins
Purdue University, West Lafayette, IN, USA
e-mail: mjellins@purdue.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_11

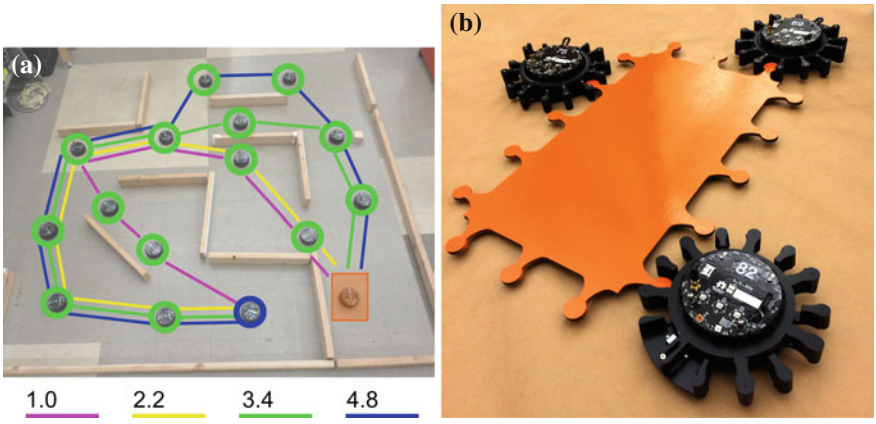
1 Introduction

Multi-robot systems offer the potential to transport many objects simultaneously. Large populations of robots offer a scalable approach; individual robots can move small objects, while cooperating robots can move large objects. A necessary requirement is to produce a path through the environment that takes into account the key dimensions of the object, and the costs of rotation and translation. This problem is especially challenging when traversing through an unknown environment where the path is undefined, and the environment might change. There have been several studies on planning the path for object transport. Kamio and Iba [6, 7] used RRT techniques to transport an object between humanoid robots. Their work used the global positioning for robots, and the environment was known. Golnzalez and Tores [15] planned a path for a box pushing problem. Their algorithm was centralized and global position information was given. Reina and Caro [16] used aerial ceiling cameras for covering the environment where the object is transported. These robots cooperatively planned the path for the object.

We are looking for a scalable and fully distributed path-planning algorithm for transporting a large object using a multi-robot system, one that cannot easily be transported by a single robot. We split the task of object transport into two tasks of path planning and object manipulation. Yamashita et al. [17] have used a similar approach, however, they used global positioning and centralized algorithms.

We are motivated by using sampling-based planning such as traditional PRM and RRT [8, 10]. These techniques provide a discrete model of a continuous path planning problem and is appropriate for distributed algorithms. But, these planners require global information. We assume the robots have no prior knowledge of the environment or other global information, mimicking real-world situations where services such as global positioning are not available or are too expensive to implement. We assume the environment is larger than the robot's communication range. We assume the robots are dispersed throughout the environment randomly, but perhaps non-uniformly. We present a fully distributed path planning algorithm that uses a large population of robots to *sample* the environment. We call these samples as *Smart Samples*. Smart samples can communicate and share some knowledge about the object as well as the information about the relative position of their neighboring samples and their closed by obstacles. We use a variation of distributed Bellman-Ford algorithm to generate the shortest path tree from the start position of the object to the goal. Previously, O'Hara and Balch [14] used a variation of distributed Bellman-Ford Algorithm [5] to plan paths for dynamic environments. However, their work focused on robots navigation and it did not incorporate the object transportation.

This paper presents a *distributed* path planning algorithm. We assume robots explore the environment with some tools [4]. Robots cover the environment and can detect the obstacles. Using local obstacle information, robots construct a configuration space for the environment in a distributed fashion. We select a robot near the goal position as the root, and a shortest-path tree, based on a custom cost function, is then constructed. The cost for the tree takes into account the amount of



Path Planning Object Manipulation

Fig. 1 **a** An environment is sampled by 16 r-one robots (*green circles*). Our goal is to generate a path for transporting the object (*orange*) to the goal location (*blue*). The different paths in various colors are generated based on different control costs when navigating through in narrow corridors. **b** Carrier r-one robots with grippers attachment can translate and rotate an object

mechanical work required to transport the object and the additional control cost to navigate in constricted narrow corridors close to obstacles. The resulting tree gives a path from any other robot to the goal robot. This path is optimal with respect to the sampling of the environment and dimension of the object. The path-planning algorithm is self-stabilizing and runs continuously to reconstruct the tree if the environment changes. This makes the algorithm robust to dynamic obstacles and changes in robot population.

We implement our algorithm on the r-one robot platform, which is a low-cost robot designed for research in multi-robot systems [13]. It has a large sensor suite and is capable of motion and local communication. We plan to develop distributed controllers and algorithms based on the r-one platform to form a complete object transport solution. We have designed and built grippers for the r-ones. Grippers enables robots to attach, rotate, and translate an object collectively as shown in Fig. 1b [13]. The detail of object manipulation will be discussed in future work. This paper focuses on the path planning aspect in an unknown environment.

2 Model and Assumptions

Multi-robot network is modeled as a undirected unit disk graph $G = (V, E)$ [2]. Each robot is represented by a node, $u \in V$, where V is the set of all robots and E is the set of all robot-to-robot communication links. The neighbors of each robot $u \in V$ are the set of robots with line-of-sight connections and within communication range

r from robot u. Robots can measure relative orientation between its heading and its neighbors' as well as their distance from its neighbors and obstacles [13]. A robot is modeled as a small disk with a pose defined by $u_{pose} = (x, y, \theta)$. Each robot is at the origin of its local coordinate system. The x-axis is aligned with the robot's current heading. The robots have a differential drive that allows them to rotate in place or translate. We model algorithm execution as proceeding in a series of discrete *rounds*. While the robots actual operation is asynchronous, implementing a synchronizer simplifies analysis greatly and is easy to implement [12].

We split robots into two groups: planner robots and carrier robots. Planner robots generate the path and carrier robots transport the object along the generated path. We design distributed controllers for carrier robots which enables them to rotate and translate the object along the path. Future work will focus on these controllers. This paper focus on planner robots and path planning algorithm. To simplify our notation, we use robot instead of planner robot in the rest of the paper. Robots require some knowledge about the object to plan the path. We simplify the object to its bounding rectangle as illustrated in Fig. 2a. We assume that the object rotate around its *center* which is the centroid of its bounding box. This simplification allows us to check the collision if the bounding rectangle collides with obstacles. Given *MaxDim* and *MinDim* of the object (see Fig. 2a), each robot measures the distance to the nearest obstacle and checks whether the object collides with the obstacle if its center is

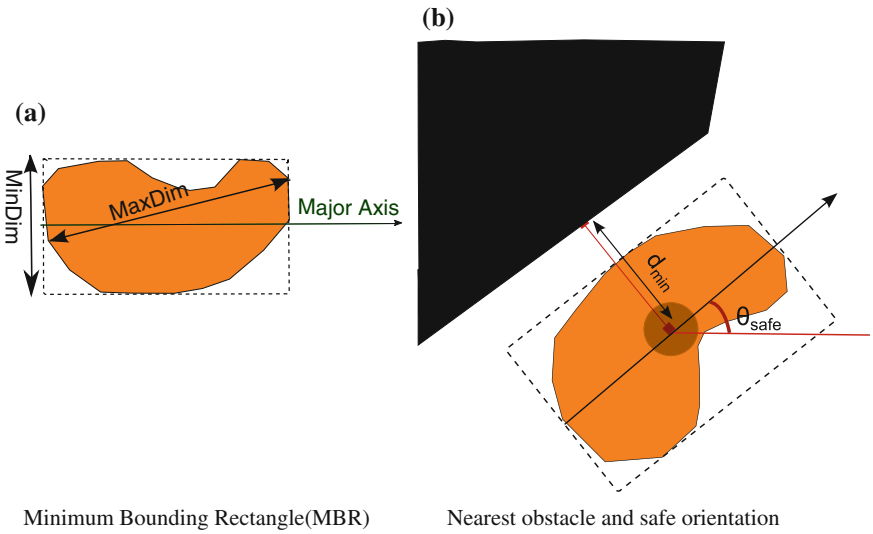


Fig. 2 **a** The object and its bounding rectangle. MaxDim is the maximum dimension of the object during rotation. MinDim is the minimum dimension of the object. The object orientation is defined as the orientation of major axis of the bounding rectangle. The object center is shown by a *blue circle*. **b** Safe orientation, θ_{safe}, for the object when it moves close to the obstacle (**SM** robots). To avoid collision, the object orientation is orthogonal to the normal of the nearest obstacle. d_{min} is the distance to the nearest obstacle and is measured by the robot (*gray circle*)

located at robot's center. Measuring *MaxDim* and *MinDim* of the object is left as our future. This paper assumes these measurement is given to the robots.

3 Distributed Path Planning

We assume there is a disperse but connected network of robots that cover the environment by using a known expansion algorithm [4]. The object is detected by the robots and the closest robot is selected as *start* robot. There is also a distinguished robot as *goal* robot at destination. We aim to generate a path among the network of robots from start robot to the goal robot. Our path planning algorithm executes in three phases. First, the robots build a configuration space in a distributed fashion. Then, they construct a shortest-path tree using a variant of the distributed Bellman-Ford algorithm from the goal robot to every other connected robot in the network. The closest robot to the object is selected as the start robot and the shortest path among the tree is constructed from start to the goal. We explain each step in detail in this section.

3.1 Configuration Space

The goal is to generate a path to carry an object. This path should be safe, meaning that the object does not collide with any obstacle during the transport. To satisfy the safety, robots require to find the safe regions in the environment and avoid to generate a path in unsafe areas. Each robot's position is considered as a potential position of the object. Since the object is simplified to its bounding rectangle, each robot checks the collision of this rectangle if the object center is located at robot's position. In order to check the collision, a robot measures its distance to the closest obstacle, called d_{min}, and determines its status along the navigation path of the object and classifies itself as one of Unsafe (U), Safe (S), or Safe Minor (SM). The classification indicates whether the object would collide with an obstacle if the object center was at the location of the robot. If the object cannot collide in any orientation, then the robot position is safe. If the object collides in every orientation, then the robot position is unsafe. If the object does not collide when its orientation, which is the direction of minor axis, is perpendicular to the normal of nearest obstacle, then the robot position is safe minor.

In global reference view, the robots independently sample the environment and construct an approximation configuration space based on the safety classifications of their locations. The configuration space is categorized into three types of safety regions: safe, safe minor, and unsafe. The classification $C(x,\ y)$ and safe orientation $\theta_{safe}(x,\ y)$ for a robot at global coordinates $(x,\ y)$ are defined according to the following conditions:

$$\begin{array}{lrcl} \textbf{if} & MinDim < d_{min} & \Rightarrow & C(x,\ y) = \textbf{S},\quad \theta_{safe}(x,\ y) = \textbf{undefined} \\ \textbf{if} & MinDim \leq d_{min} < MaxDim & \Rightarrow & C(x,\ y) = \textbf{SM},\ \theta_{safe}(x,\ y) = f(x, y) \\ \textbf{if} & d_{min} < MaxDim & \Rightarrow & C(x,\ y) = \textbf{U},\quad \theta_{safe}(x,\ y) = \textbf{undefined} \end{array}$$

In this classification θ_{safe} is the safe orientation of the object in order to avoid the obstacles. θ_{safe} is undefined for **S** and **US** regions. In **S** regions the object can rotate freely with no risk for collision. There is no need for a specific orientation. Similarly, in **U** regions the object collides in any orientation so θ_{safe} is also undefined. However, in **SM** regions, the orientation of the object is constrained to a range of angles. We define $f(x,\ y)$ as a function that computes θ_{safe} in **SM** regions. To be conservative, we limit θ_{safe} to be orthogonal to the normal of the nearest wall as shown in Fig. 2b. Orienting the object at this angle maximizes the distance between the object and the wall by making the MinDim side of the bounding box parallel to the normal of the wall.

Figure 3 shows an environment sampled by a large number of infinitesimally small robots achieving an infinite resolution configuration space. The configuration space can be segmented using Voronoi tessellation of edge and vertices of the obstacles [3, 11] into regions that share the same closest obstacle. Combining the two, we can visualize the SM regions that have the same safe orientation of θ_{safe}. The individual robots have no global information about the environment and thus cannot rely on this method. Instead, each robot finds the closest obstacle edge or vertex, classifies its safety, and sets θ_{safe} individually. If robot's view is blocked by other robots, robot uses distance and bearing information of its neighbors, and computes its θ_{safe} in its own coordinate system.

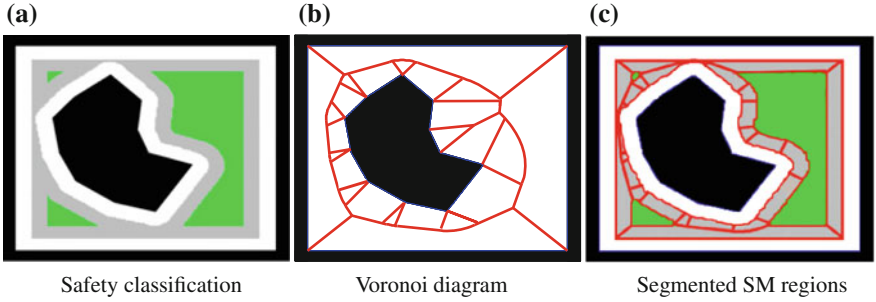


Fig. 3 **a** The configuration space constructed by an infinite number of robots. The locus of **S**, **SM**, and **U** points are *green*, *gray*, and *white* respectively. The walls and obstacles are *black*. **b** Voronoi tessellation separates the configuration space into regions based on the location of the closest obstacle, edge or vertex. **c** Combining configuration space and Voronoi tessellation, we split the **SM** regions into sections with the same safe orientation θ_{safe}

3.2 Distributed Bellman-Ford Algorithm

We model path planning problem as finding the shortest-path problem in a directed graph with single source at goal location (Fig. 4a). The cost of transport from node u to node v, which is the weight of edge $w(u,\ v)$, is calculated based on the mechanical work required to translate and rotate the object from node u to neighbor node v. Note that $w(u,\ v)$ and $w(v,\ u)$ are not necessarily the same. This happens when translating between **S** and **SM** locations. The weight is defined by the following equation:

$$w(u,\ v) \ = \ \alpha c_f d(u,\ v) + c_\tau \theta_\Delta \tag{1}$$

where $d(u,\ v)$ is distance between nodes u and v. θ_Δ is the change in angle for the object to be orientated at the safe heading θ_{safe}. c_f and c_τ are constants of force and torque that scale $d(u,\ v)$ and θ_Δ to the mechanical work required to translate and rotate the object respectively. These coefficients depend on the dimensions and weight of the object. α is called *minor translation control multiplier*, an additional weight for extra control cost when the object is being transported in confined regions (**SM** locations). α is an user-defined value and is ≥ 1 when moving to a **SM** region, where the object's motion needs to be controlled more tightly to avoid collision. A rotation cost is also imposed to orient the object to θ_{safe}. When moving to a **S** region, the object can rotate freely without collision, the α is 1 and rotation cost is 0. We use the worst case θ_Δ (change of object's orientation) of $\frac{\pi}{2}$ when moving from a **S** region to a **SM** region's θ_{safe}. The cost function for all different safety region transitions is summarized by Table 1.

We introduce *safety factor* γ to quantify safety. This measurement is useful for analyzing the path planning performance and check if the path is generated in narrow passages. The larger the γ is, the farther the object is going to be transported away from the obstacles. In S and SM regions $\gamma \geq 1$.

$$\gamma \ = \ \frac{d_{min}(x,\ y)}{MinDim/\ 2} \tag{2}$$

To quantify the efficiency of the generated path, we define *path efficiency* as the ratio of the shortest path length to the planned path length.

Table 1 Translation and rotation cost for different safety regions

Edge type	Translation cost	Rotation cost
S → S	$c_f d(u, v)$	0
SM → S	$c_f d(u, v)$	0
S → SM	$\alpha c_f d(u, v)$	$c_\tau \frac{\pi}{2}$
SM → SM	$\alpha c_f d(u, v)$	$c_\tau \theta_\Delta$

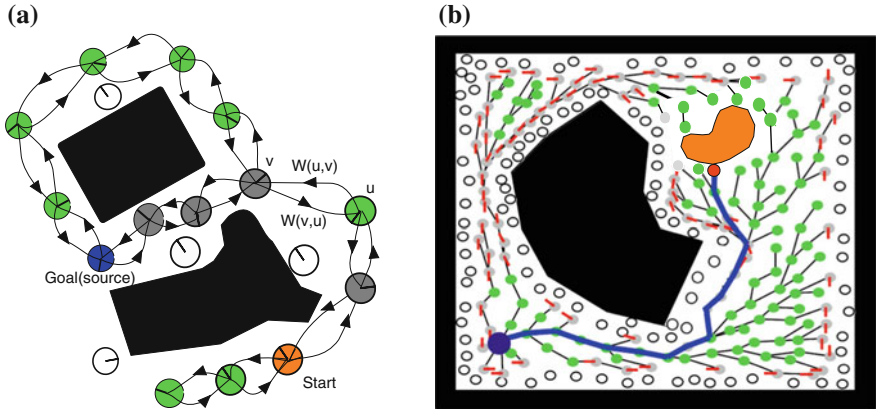


Fig. 4 **a** A directed graph as the model for the robot network. Robots in **S**, **SM**, and **U** are shown in *green*, *gray*, and *white* respectively. Goal is *blue* and start is *orange*. W(u,v) and W(v,u) are the cost of moving the object from node u to v and moving from v to u respectively. **b** The optimal path (*blue*) is found using a variant of the distributed Bellman-Ford algorithm for a simulated environment sampled by 275 robots. The goal robot is indicated by a *large blue circle*. Start point is the robot in red, which is the closest robot to the object (*orange shape*). All other robots are designated by *small circles* and are classified as S (*green*), SM (*gray*), and U (*white*). *Red lines* on SM robots denote the safe orientation for the object at that location.

$$\psi = \frac{P_{shortest}}{P_{planned}} \tag{3}$$

where $P_{shortest}$ is the sum of all edges for the shortest distance path. $P_{planned}$ is the sum of all edges for the path planned by our algorithm.

We use a variant of the distributed Bellman-Ford Algorithm [1, 5] to build the shortest-path tree. We use the above cost function to find a minimum cost path from any **S** or **SM** connected robot in the network to the goal position. **U** robots are not traversable locations and are excluded in the tree. We assume that there is a sufficiently large population of robots that are scattered to sample the environment and there is at least one path from the starting location to the goal. We also assume that robots sample the environment so efficiently that they detect all the edges and vertices of the obstacles.

Figure 4b shows the result of our algorithm in a simulated environment. Robots categorize themselves into three types of safe, safe minor and unsafe in a distributed fashion. Safe minor robot are oriented along with safe orientation. The shortest path tree is built through **S** and **SM** robots. Path is built continuously which allows changes in robots network topology.

3.3 Transporting the Object

After the minimum-cost path-tree is generated, the object transport starts from any location on the tree and is transported by carrier robots. When the object moves to a **SM** location, the object is rotated until it is oriented to the safe orientation of that location (θ_{safe} or $\theta_{safe} + \pi$ for an undirected object). We design distributed controllers to rotate and the translate the object. The prescriptions of the controller is outside of the scope of this work, but preliminary results show that translation, rotation, and combined controllers can successfully transport an object by carrier robots. Physical interference from the robots will be a significant problem and is left for later work.

4 Experimental Results

We tested our path planning algorithm in a simulated environment. Figure 5 shows a complex environment sampled by a large number of robots. Using our algorithm, different paths are generated by altering the minor translation control multiplier α. From left to right, images show the change of the path by increasing (α). The path becomes longer due to inclusion of safe and less safe minor robots. We can balance the trade-off between safety and overall path length by adjusting α. Path efficiency ψ and safety factor γ for this environment are plotted in Fig. 6. As we expect, path efficiency decreases and the safety factor increases by increasing α.

Basically, our algorithm is a sampling path planning in which robots sample the environment. Similar to other sample-based path planning, the path efficiency directly depends on the number of the sample(robots). Figure 6 shows the effect of network degree on path efficiency. By increasing the degree of the network, which implies network size, the path efficiency approaches to the unity. For the degree larger than 12, the environment is sampled enough so that our algorithm generates the path close to the shortest path. The population with degree < 5 is disconnected

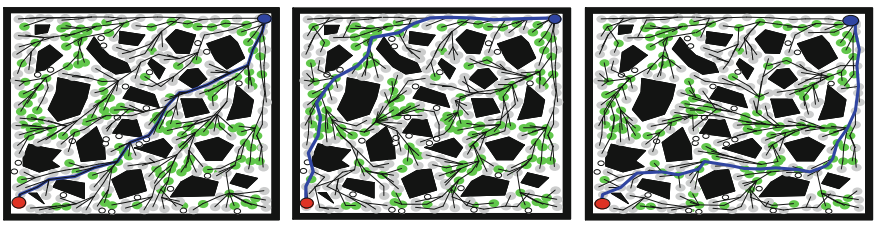


Fig. 5 Simulation of the path planning algorithm for an environment sampled by 339 planner robots. Robots have safety designation of **S** (*green*), **SM** (*gray*), and **U** (*white*). Start and goal positions for the object are marked *red* and *blue* respectively. The *blue line* shows the planned path found by our algorithm. By altering α in the cost function in the simulation from *left* to the *right*, the algorithm generates different topology for the shortest-path tree and the final path changes

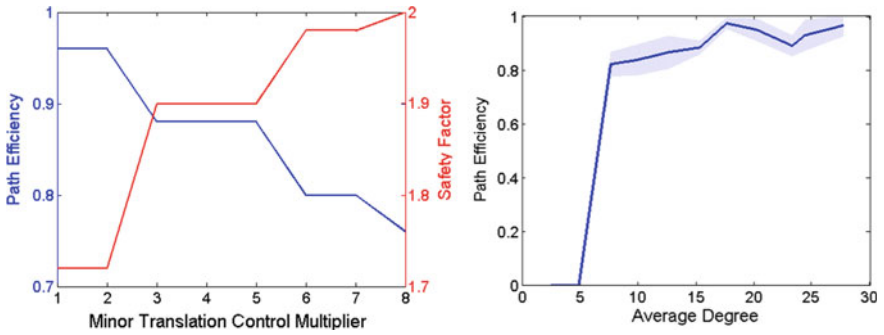


Fig. 6 *Left* Plot for path efficiency ψ (*blue*) and safety factor γ (*red*) with respect to minor translation control multiplier α for the environment used in the simulation of Fig. 5 with 1000 robots. *Right* The effect of network degree on the average of path efficiency of algorithm in the environment used in Fig. 6, with minor translation control multiplier $\alpha = 1$, the result is for the average path efficiency of 10 networks for each degree

and the algorithm cannot find any path from start to the goal and path efficiency is zero. For careful analysis of this behavior see the work by Kleinrock [9].

We conducted several experiments using the r-one robots. They have knowledge of their pose and their neighbors' in their local coordinate frames. They communicate with their neighbors through line-of-sight infrared transmitters and sensors (we tune communication power to give maximum range of 0.375 m). Robots also sense nearby obstacles with infrared ranging to find their safety classifications. The **SM** robots orient themselves to their respective θ_{safe} angles based on the nearest obstacle. This aids debugging, and simplifies the coordinate systems. For the experiments, the cost function is simplified. We assume each robot has a distance of 1 form its neighbors and the distance-work scaling factor c_f is 1 (i.e. $c_f d(u, v) = 1$).

We constructed three large and complex environments with multiple paths from start to goal in Fig. 7. Each row shows the algorithm running with different minor translation control multiplier values. These values increase from left to the right. By measuring the path distance and the average distance for all robots on the path from the nearest obstacle, we are able to measure the path efficiency, ψ, and the safety factor, γ for each path. The data can be found in Fig. 8. Similar to Fig. 6, ψ decreases and γ increases with α increases.

Our algorithm is self-stabilizing, meaning that it is robust to changes of the network topology. We test self stabilization property in a dynamic environment by constructing an environment with a "room" and two "doors" that lead to the destination (Fig. 9). The doors serve as movable obstacles. Our algorithm first selects the shortest path when the doors are open. After the environment is altered and a door is closed, the initial path is destroyed. The algorithm successfully regenerates an alternate path to the goal. This Experiment demonstrates our algorithm's ability to respond to evolving network topology and to regenerate the path when the initial path is broken.

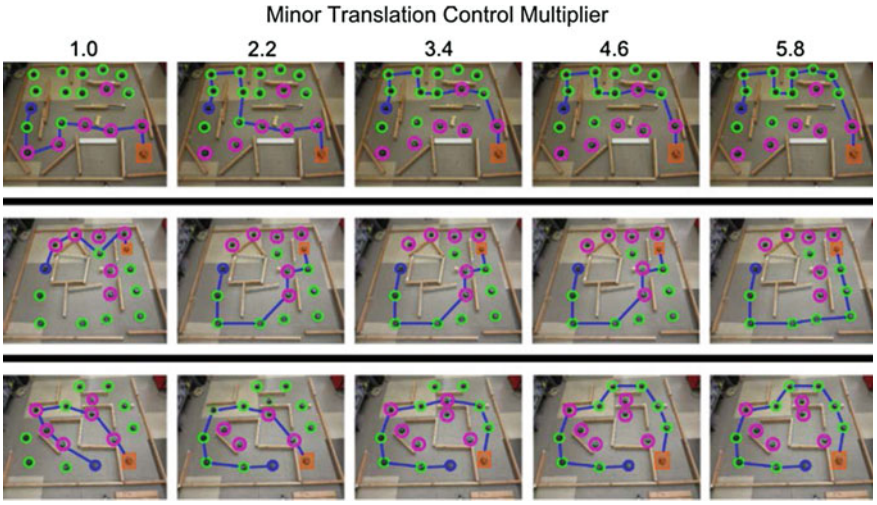


Fig. 7 The path planning algorithm tested on three different complex environments with a varying minor translation control cost α for each. **S** and **SM** robots are labeled *green* and *magenta* respectively. The dimensions of the environments are 2.6×2.6 m. Goal robot is *blue* and starting robot is *orange*. α increases from *left* to *right*. The path changes with increasing α value to include fewer **SM** robots for a safer transport

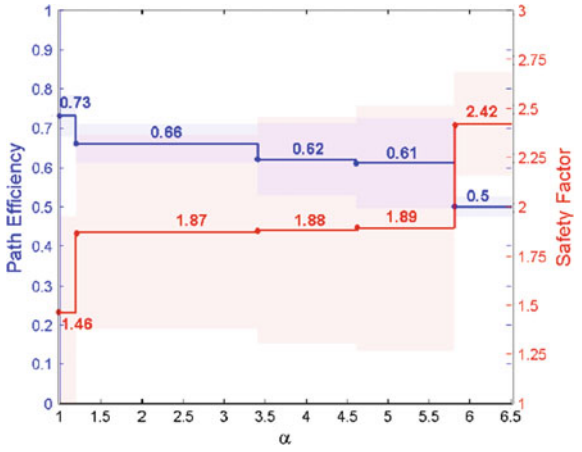


Fig. 8 The result of path efficiency (*blue*) and safety factor (*red*) are plotted against minor translation control multiplier. The data was gained from the experiments shown in Fig. 7. The region around each plot shows standard deviation from the mean for the three tested environments

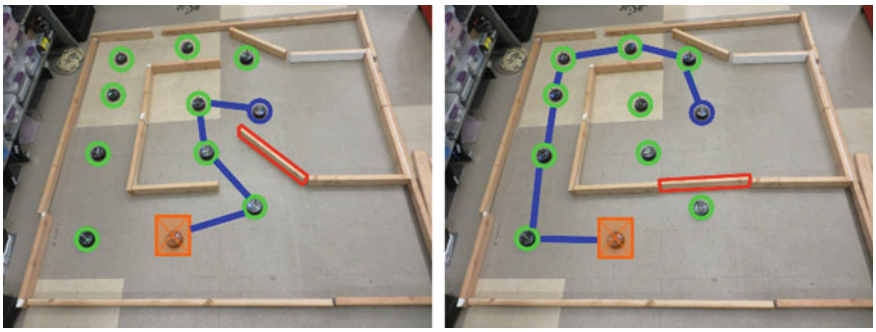


Fig. 9 Experiment with 11 robots. Goal robot is shown in *blue*. The object bounding rectangle is *orange*. Other robots are *green*. At first, a door (*red rectangle*) was open (*left*) and the shortest path through the robots is generated (*blue line*). When the door is closed (*right*) the initial path is broken and algorithm find an alternative path

5 Scope and Limitations

The path planning algorithm is fully distributed and is robust to changes in the environment and the robot network topology. We can balance the trade-off between safety and efficiency by tuning parameter α. Since the algorithm is a variant of distributed Bellman-Ford algorithm, it has a running time complexity of $O(diam(G))$, where $diam(G)$ is the diameter of the graph.

Our models and algorithms have made some simplifications and assumptions to make the path planning computationally feasible. However, they do have limitations. Figure 10 shows a case that the path between two **SM** robots is not safe, while it is selected by algorithm. One solution to fix this issue is to remove the path that connects two **SM** robots with different safe orientation constraints. However, the algorithm is conservative and may not be able to find any solutions. This problem could be addressed in future work by adding additional rotational metrics when constructing the configuration space. The path given by Fig. 10 should be disconnected due to rotational constraints. Alternatively, the model for the object could be modified into a larger rectangular enclosing box. While this method would not solve the problem completely, it could alleviate the degree of the problem at the expense of efficiency.

6 Conclusion and Future Work

We designed a fully distributed path planning algorithm to generate a path for an object transport task. We use a large number of homogeneous robots to sample an unknown environment and construct a configuration space based on safety of object constraints. We then used a variant of the distributed Bellman-Ford algorithm with a custom cost function to find a path for an object to be transported from the start to the goal location. We implemented our algorithm both in simulated and real-world

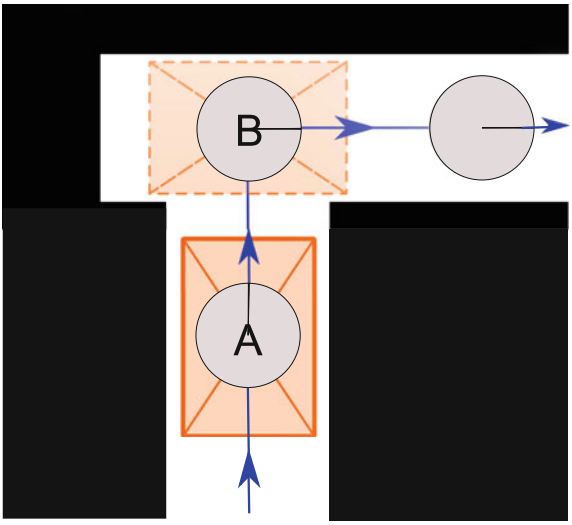


Fig. 10 The path between two nodes A and B is not safe, although the object can be safe at each of these positions if it is oriented as illustrated

environments. As our results show, our approach robustly constructs safe paths and adapts to dynamic environments. We have also investigated a path planning problem that provides a framework for the user to balance the trade-off between safety and mechanical work required to move the object using a multi-robot system.

The proposed path planning algorithm simplifies many aspects of the transportation. A more extensive study could be done in the future to model the system in greater details. The manipulation and transportation of the object are still left for future investigation, although preliminary results are promising. The object dimension and orientation should be measured dynamically by the manipulator robots and propagate through the network. Additionally, moving the object may cause physical interference. We will address these issues in our future work.

In conclusion, our proposed path planning method is the first step for an effective object transportation using a multi-robot system. More research still need to be done in order to utilize it to its full capacity.

Acknowledgments The authors would like to thank Zachary Kingston for his tremendous help in running experiments on real robots. This work has been supported by National Science Foundation, Division of Computer and Network Systems under CNS-1330085.

References

1. Cheng, C., Riley, R., Kumar, S.P.R, Garcia-Aceves, J.J.: A loop-free extended Bellman-Ford routing protocol without bouncing effect. In: SIGCOMM '89 Symposium Proceedings on Communications Architectures and Protocols, vol. 19, pp. 224–236 (1989)
2. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discr. Math. **86**(1–3), 165–177 (1990)

3. Fabbri, R., Estrozi, L.F.: On Voronoi diagrams and medial axes. J. Math. Imaging Vis. **17**, 27–40 (2002)
4. Fekete, S.P., Kamphans, T., Kröller, A., Mitchell, J.S.B., Schmidt, C.: Exploring and triangulating a region by a swarm of robots. In: 14th International Workshop, 2011, and 15th International Workshop, pp. 206–217, Princeton, NJ, USA (2011)
5. Ford, L., Fulkerson, D., Bland, R.: Flows in Networks, ser, Princeton Landmarks in Mathematics. Princeton University Press, Princeton (2010)
6. Kamio, S., Iba, H.: Random sampling algorithm for multi-agent cooperation planning. In: IROS, pp. 1265–1270. IEEE (2005)
7. Kamio, S., Iba, H.: Cooperative object transport with humanoid robots using rrt path planning and re-planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2608–2613. IEEE (2006)
8. Kavraki, L., Svestka, P., claude Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In: ICRA, pp. 566–580 (1996)
9. Kleinrock, L., Silvester, J.: Optimum transmission radii for packet radio networks or why six is a magic number. In: Conference Record, National Telecommunications Conference, pp. 4.3.2–4.3.5, Birmingham, Alabama, Dec 1978
10. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Computer Science Department, Iowa State University, Technical report (1998)
11. Mayya, N., Rajan, V.T.: Voronoi diagrams of polygons: a framework for shape representation. J. Math. Imaging Vis. **6**(4), 355–378 (1996)
12. McLurkin, J.: Analysis and implementation of distributed algorithms for multi-robot systems. Ph.D. dissertation, MIT, USA (2008)
13. McLurkin, J., McMullen, A., Robbins, N., Habibi, G., Becker, A., Chou, A., Li, H., John, M., Okeke, N., Rykowski, J., et al.: A robot system design for low-cost multi-robot manipulation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pp. 912–918. IEEE (2014)
14. O'Hara, K.J.O., Balch, T.R.: Distributed path planning for robots in dynamic environments using a pervasive embedded network. In: Proceedings of the Thrid International Joint Conference on Autonomous Agent and Multiagent Systems, pp. 1538–1539, July 2004
15. Parra-González, E.F., Ramírez-Torres, J.G., Toscano-Pulido, G.: A new object path planner for the box pushing problem. In: Electronics, Robotics and Automotive Mechanics Conference: CERMA'09, pp. 119–124. IEEE (2009)
16. Reina, A., Di Caro, G.A., Ducatelle, F., Gambardella, L.M.: Distributed motion planning for ground objects using a network of robotic ceiling cameras. In: Towards Autonomous Robotic Systems, pp. 137–148. Springer (2011)
17. Yamashita, A., Arai, T., Ota, J., Asama, H.: Motion planning of multiple mobile robots for cooperative manipulation and transportation. IEEE Trans. Robot. Autom. **19**, 223–237 (2003)

Collective Construction of Dynamic Equilibrium Structure Through Interaction of Simple Robots with Semi-active Blocks

Ken Sugawara and Yohei Doi

Abstract This paper proposes a collective construction method through interaction between simple robots and intelligent blocks that has a rule set and functions to communicate with neighboring blocks. In our proposed method, the structure is formed by growing chain of blocks. The growth direction is determined by the rule set and a counter value passed between the blocks. The robots load or unload the block based on a simple algorithm and a local signal from the blocks. Because of the simplicity of each robot's behavior, the structure is locally unstable: the blocks can be attached or detached randomly from the structure even if they have already formed a part of the structure. The structure is considered a dynamic equilibrium structure that is locally unstable but globally stable. In this paper, we first explain the mechanism of our proposal and show some fundamental characteristics obtained by computer simulation. Then, we show the adaptability of the system by introducing simple sensing dynamics for an external stimulus.

Keywords Collective construction · Dynamic equilibrium · Swarm robots · Semi-active blocks

1 Introduction

Collective construction is one of the most attractive and challenging topics in swarm robotics. Construction is a practical and indispensable task for humans, and we often desire that robots will provide support or work fully instead of us. Construction by robots is expected to improve productivity, and to reduce accidents in the construction field. Collective construction by swarm robots is also promising. Once we succeed in

K. Sugawara (✉) · Y. Doi
Tohoku Gakuin University, Sendai, Japan
e-mail: sugaken@mail.tohoku-gakuin.ac.jp

Y. Doi
e-mail: yohei.doy@gmail.com

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_12

establishing collective construction by swarm robots, we expect our method to work effectively not only in a normal environment, but also in an extreme environment that is difficult to reach, such as deep sea or other planets.

We can classify construction into two types of management approaches: *centralized* and *distributed*. In centralized management, we have a complete set of blueprints, and manage all workers in every assembly process. Artificial buildings are constructed by this method. The working efficiency is relatively high because everything is completely controlled by a supervisor. We know, however, that another method is also possible: distributed management. Social insects, such as bees, ants, and termites, can construct a large and complex nest using the distributed method. Although these insects seem to have neither clear blueprints, nor a central supervisory control system, they can still construct a functional nest.

It is challenging to solve the collective construction problem using a distributed system because we can expect to get a novel methodology for the problem. Once we succeed in establishing the problem based on the distributed approach, we can increase the variety of collective construction methods for our use. In this paper, we propose a novel method for collective construction by introducing a simple dynamics for each robot and each block, and discuss the characteristics of the formed structure by considering the dynamic equilibrium phenomenon.

1.1 Related Work

The algorithm for constructing a cluster proposed by Deneubourg et al. [1] was epoch-making in the research of distributed autonomous system. In spite of a simple algorithm, distributed items are collected automatically. The algorithm's feasibility was also confirmed by a real robot system [2]. The idea was extended to the sorting problem, in which the extended method sorted and clustered a mixture of different items. Its practicability was confirmed by robotic implementation [3], and by clustering tasks in database engineering [4].

There are other approaches for robotic construction. By simple physical contact between the robot and items, it was confirmed that the robots organized a heap of items [5] or created a circular area by pushing the items away [6]. In addition, introducing a beacon system allowed construction of more complicated structures [7, 8].

Stigmergy is also a key concept for collective construction [9]. Some researchers proposed *simple dynamics*, which explains how insects construct their nests autonomously [10, 11]. Theraulaz and Bonabeau also reported an epochal model [12, 13]. They proposed an asynchronous automata model with some micro-rule sets, and showed it could generate various types of structures, which resemble to real nests constructed by insects, such as bees and ants.

Concerning robotics, another series of researches inspired by termite nest construction shows great promise [14–16]. These researches show a method of using simple rules to construct not only a two-dimensional, but also a three-dimensional structure; the researches also showed that real robots can construct the structure by bringing from the field tand piling up the blocks.

1.2 Contribution of This Paper

We propose a novel method for collective construction of some geometric structures, introducing simple dynamics for swarm robots and semi-active blocks. We focus on the behavior of a collection mechanism that was reported by Deneubourg et al. through the observation of real ant behavior. They reported a simple algorithm, in which each robot stochastically loads or unloads an item based on the density of items around the robot. This algorithm is remarkable as the simplest algorithm for item collections by simple robots. This paper, however, reveals another aspect of this phenomenon. Once we focus on the position of each item in this dynamics, we notice that the relative positions of the items keep changing dynamically. Even though all items gather as a cluster, each item is always attached or detached locally. In other words, the cluster is in the state of *dynamics equilibrium*. Some interesting research for collective construction has been reported in the field of swarm robotics; most research has treated static structures and only a small amount of research has treated the collective construction from this viewpoint [16]. We studied this problem using a simple algorithm for the robots based on Deneubourg's algorithm and the blocks which has a *rule, internal value*, and a *limited communication method*. In order to form a designed structure by a simple system, not only the robots but also the blocks should have some intelligence [14]. Here, the robots and blocks interact and assemble a structure globally, but the blocks that compose the structure are always moved in and out locally. This characteristic also enables the structures to have fault-tolerant characteristics or high adaptability for environmental changes.

We discuss fundamental properties of the proposed system in the next section. We then provide workability of our proposal by showing some results obtained by computer simulation in Sect. 3. This section also describes simple analysis by master equation. In Sect. 4, we discuss the adaptability of the structure to environmental change. The discussion and conclusion are in the last section.

2 Model

Our model is composed of two parts. One is for robots, and the other is for blocks. First, we show an algorithm for the robots, and next, we explain the dynamics of the blocks.

2.1 Algorithm for a Robot

Deneubourg et al. proposed a simple clustering model based on stochastic dynamics for picking up or depositing the items. The simplest algorithm of their proposal can be described as follows (Algorithm 1).

Algorithm 1

```
LOOP
    DO random walk UNTIL find a block

    IF holding no block THEN
        Pick up the block
    ENDIF

    ELSE IF holding a block THEN
        Drop off the holding block close to the found block
    ENDIF
```

We extend this algorithm as shown in Algorithm 2. As described below, each block is assumed to have a function to send signals and to indicate the side on which the next block should be dropped off.

Algorithm 2

```
LOOP
    DO random walk UNTIL find a block

    IF holding no block THEN
        Pick up the block
    ENDIF
    ELSE IF holding a block THEN
        IF detect signal from the found block THEN
            Drop off the holding block at the signal source
        ENDIF
    ENDIF
```

Figure 1 shows a schematic of our proposal, using an example of a robot's behavior. As the robot that is not holding a block finds one, it picks up the block. When it finds another block by a random-walk and detects a signal from it, the robot puts the held block on the surface of signal source.

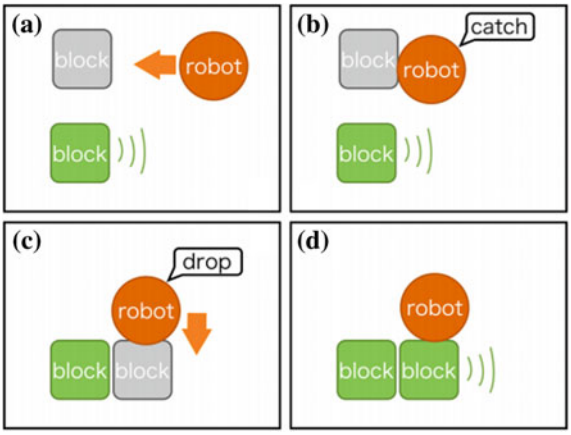


Fig. 1 Schematic of our proposal. **a** Each robot walks around randomly. **b** A robot in the unloaded condition picks up the found block. **c** The block is dropped off where the robot detects the signal. **d** The laid block starts to transmit a signal based on the neighbor's information (see detailed explanation in the next section)

2.2 Dynamics of a Block

Blocks introduced in this research are not passive items but rather semi-active ones, which have a function to communicate with robots and/or other blocks. This concept was already proposed in [14]. In this paper, we assume all blocks have a set of rules, memory for a counter value, and a minimal communication method. The block introduced in this paper can indicate the side on which another block should be placed.

The rule of our block is composed of trigger values, growth directions, and a new counter value as an option. For simplicity, we assume the grid world in this paper, in which a series of blocks has seven growth directions with various combinations of the directions. The following symbols indicate the side on which the next block should be placed.

S: straight from the former block
DL: diagonally forward left
L: left
DR: diagonally forward right
R: right
DBL: diagonally backward left
DBR: diagonally backward right

It is also possible to indicate two or more directions in parallel, meaning that a block can communicate from two or more sides at once; hence, its growth can branch out.

We introduced the counter to control the length of the growth. For example, when we design a branch with ten blocks, the counter is set as 10. In other words, this system needs to have an origination.

Here we define the rule set as {initial count value: (trigger value, direction, (new value, as an option))}. The following examples illustrate the rule set process.

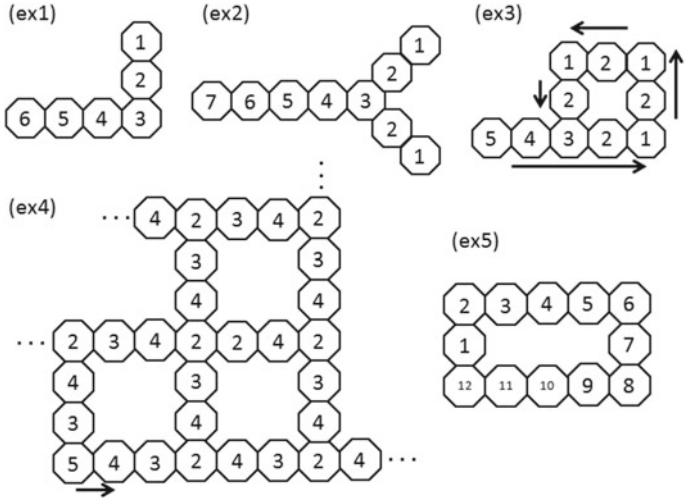


Fig. 2 Constructed structures of examples in the text. The *numbers* show the counter value in each block

(ex1) A rule set {6: (3, L)} generates an L shape
(ex2) A rule set {7: (3, DL), (3, DR)} generates a Y shape
(ex3) A rule set {5: (1, L, 2)} generates a q shape
(ex4) A rule set {5: (2, S, 4), (2, L, 4)} generates a lattice
(ex5) A rule set {12: (8, L), (6, L), (2, L)} generates a rectangle

Each block also has the following rules.

- A single block does not transmit a growth direction signal. Even if a block composes a part of the structure, it stops transmission and clears its counter value once it becomes single.
- The block in the structure does not change its communicating side. Even when another branch grows and makes contact with a part of the structure, the contacted block does not accept the signal from the newcomer.

Figure 2 shows the concrete structures described in (ex1)–(ex5).

2.3 Dynamic Equilibrium Structure

Because the robots do not care if a found block organizes a part of the structure, they pick up the block when they are in *unloaded condition* (Fig. 3). Even if there is a lack of blocks, another robot in *loaded condition* may reach the point and place another block again. This system essentially has a local instability: The blocks are not guaranteed to stay in the structure. All blocks can be attached to or detached from

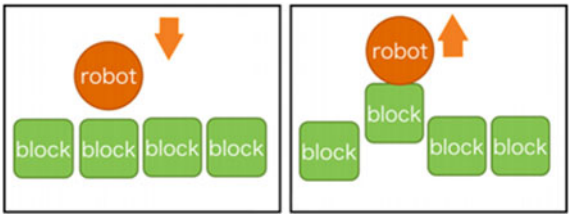


Fig. 3 One possible loading process. Because the algorithm of each robot is simple, the robot easily happens to pick up the block that forms the structure. This behavior plays an important role in our proposal

Table 1 Simulation condition

Simulation field	Discrete (lattice) space
Field size	300×300 pixels
Neighborhood	Moore neighborhood (Eight surrounding cells are available for robots and blocks)
The number of robots	5–300
The number of blocks	100, 400, 500

the structure, even if the block forms a part of the structure. However, the structure is stable because of the rule: The rules of blocks guarantee stability of structure formation.

3 Results and Analysis

We confirmed the behavior of our proposal by computer simulation. The simulation conditions are shown in Table 1.

Figure 4 shows a snapshot of two examples of construction. White dots are single blocks and red dots are the blocks in the structure.

As described in Sect. 2.3, the blocks in the structure can be pulled out. If the number of blocks in the field is fewer than the required number, holes appear in the structure (Fig. 5a). The holes are also unstable. They appear and disappear dynamically. If the number of blocks is greater than the required number, the structure becomes stable. Once the structure is completed and all robots pick up unused blocks, there is no chance that the robots will pick up the block in the structure (Fig. 5b).

We also tried to analyze a basic characteristic of structure growth by introducing the dynamics of the states of the robots and the blocks in a line structure of length L. Let us denote the number of the robots that are holding blocks as R_H and the robots that are not holding blocks as R_E. The number of blocks that compose the line structure is denoted as B_L. We describe the dynamics of each number as follows.

$$\begin{aligned} \dot{R}_H &= \alpha R_E - \beta R_H - \mathrm{p} R_H \\ \dot{R}_E &= -\alpha R_E - \beta R_H - \mathrm{p} R_H \\ \dot{B}_L &= \beta R_H - \varepsilon (L - B_L) R_H - \gamma \cdot B_L \cdot R_E \end{aligned} \tag{1}$$

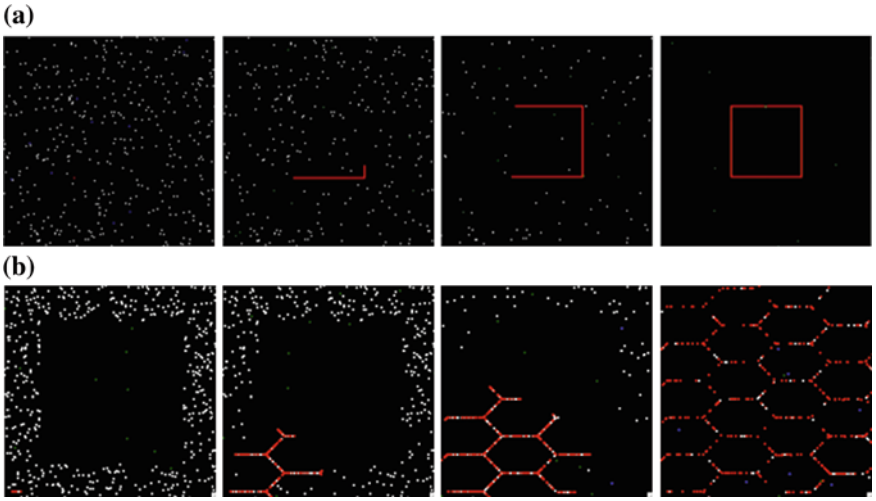


Fig. 4 Examples of construction of structure. **a** *Rectangle* **b** *honeycomb*

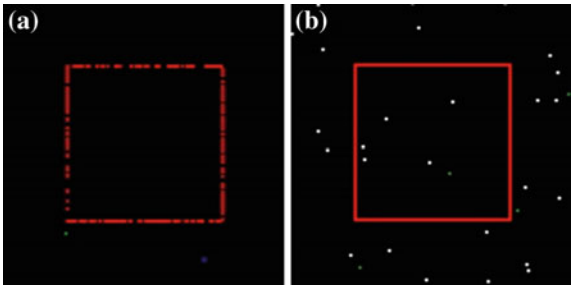


Fig. 5 **a** In case the total number of blocks is less than the required number for completion, the global structure is maintained as a *square* but there are many *holes*. **b** In case the total number of blocks is much greater than the required number for completion, the global structure is stable without holes

where α is the probability to pick up the block; β is the probability to put down the block; p is a probability to drop off the block; ε is a rate that robots holding blocks put down their block as a part of the structure; L is the length of the structure to be completed; and γ is the probability that the blocks forming the structure are detached by the robots that are not holding blocks. Figure 6 shows the time evolution of the length of the structure in robot simulation (left) and the solution of the equations described previously. Here we set $\alpha = 10^{-3}$; $\beta = \gamma = 2.5 \times 10^{-5}$; $p = 5.0 \times 10^{-4}$; and $\varepsilon = 1.3 \times 10^{-4}$.

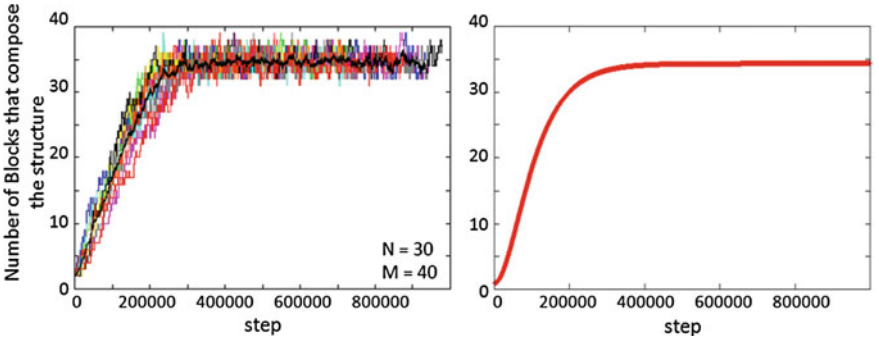


Fig. 6 (*Left*) Time evolution of the length of the structure, showing the result of 10 trials (*color lines* except the *black*) and their average (*black*). The designed length is 40 and the number of robots is 30. (*Right*) the solution of the equations. The length of the designed structure is 40 ($L = 40$)

4 Extension for Adaptive Behavior

Another interesting aspect of the system that forms a dynamic equilibrium structure is its adaptability to environmental changes. In this section, we introduce small modifications to the blocks and attempt to implement adaptive function. For example, we treat a line structure. We also assume an external stimulus, such as a strong wind or a flow of hazardous substances; the line works as a barrier against the stimulus by being constructed to be orthogonal to the direction of the wind or other stimulus.

We introduce a sensor to each block and control the flow direction of the counter value. In this case, each block detects the direction of an external stimulus and receives or transmits its counter value to the orthogonal direction. Figure 7 shows a concept of this application. A flow direction of counter value is lateral as long as the stimulus comes from the top (Fig. 7a). Once the direction changes, as shown in Fig. 7b, however, each module stops transmitting its counter value to the lateral direction and starts transmitting in the vertical direction. The lateral communication is essentially abandoned. Figure 7c shows the structure just after the direction of the external stimulus changes from the top. The blocks stop their communication for the lateral direction and starts to transmit signal to the longitudinal direction. As a result, other blocks are attached to the vertical direction. Because the anchor block in which the counter value is encircled in red in Fig. 7c, d is fixed, the leftmost column can grow and keep the line structure. However, another column will finally disappear.

Figure 8 shows snapshots of the adaptive behavior for environmental change. Once the direction of the external stimulus changes, the line structure gradually changes the direction of construction.

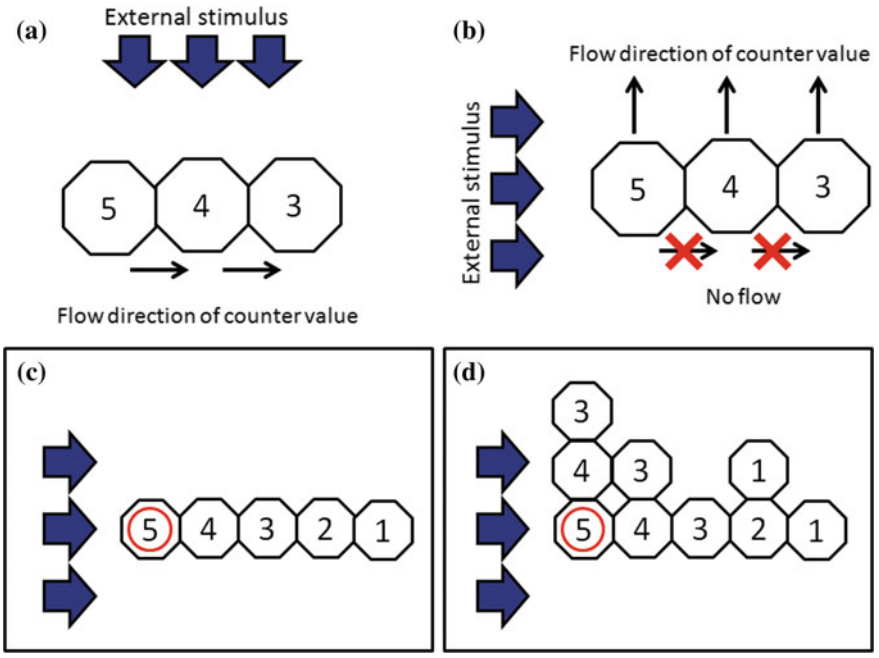


Fig. 7 **a** Each block is assumed to have a sensor to detect external stimuli and a property to flow the counter value to the orthogonal direction. **b** Once the direction of the external stimulus changes, the flow direction of the counter value also changes. **c** There is no essential connection between the blocks because of the direction of the external stimulus. **d** Line structures can grow in the vertical direction, but the line without the anchor block will be removed

5 Discussion

This system also has the potential to realize a *moving shape* (Fig. 9). This figure shows that a ribbon-like structure seems to move in an oblique direction. By allowing the anchor to be removed by the robots, we can realize a moving object. We can also say that the structure itself is regarded as a robot, which is maintained by dynamic equilibrium [17]. We need a more detailed discussion for this aspect, but we are confident that this characteristic leads to a proposal for a new type of robotics.

6 Conclusion

We propose a novel method for collective construction, which is accomplished by the interaction between simple swarm robots and semi-active blocks. Each robot works using an extension of the fundamental clustering algorithm, in which it picks up or drops off the block according to local conditions. The blocks are not passive but semi-

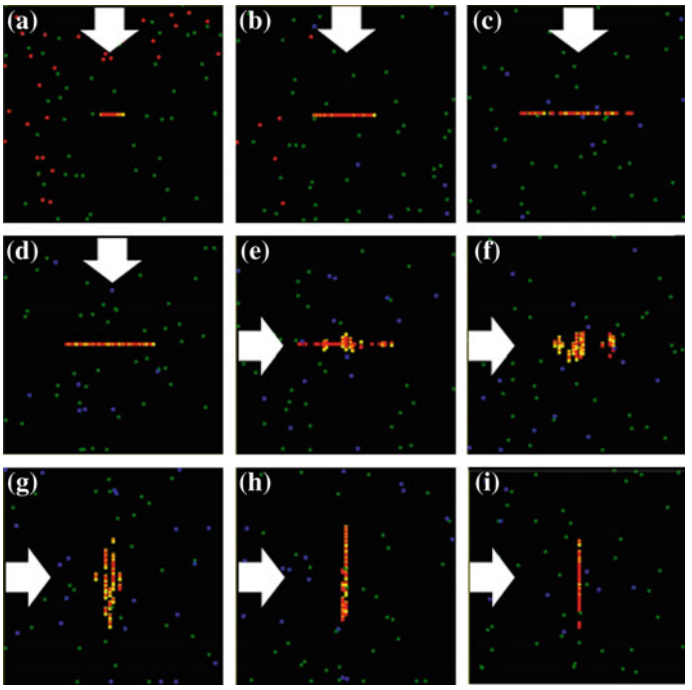


Fig. 8 Snapshots of the adaptive behavior. *Red dots* and *yellow dots* represent the blocks and the indications from the neighboring blocks for unloading, respectively. *Blue dots* and *green dots* represent the robot with a block and the robot without a block, respectively. As the external stimulus (a *white arrow*) firstly comes from the *top* (**a**–**d**), a line structure is formed in the lateral direction. Once the direction of external stimulus changes, each block attempts to grow in the vertical direction, but only one line structure which contains the anchor block remains

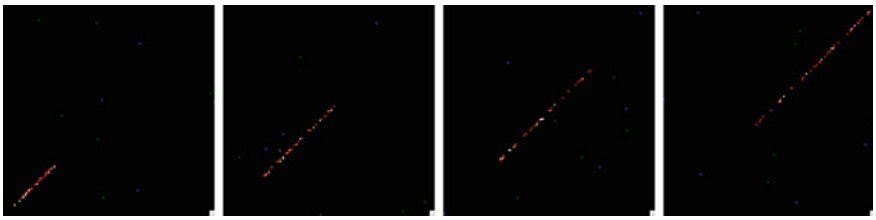


Fig. 9 Snapshots of moving structure. A line structure moves in the diagonal direction

active, that is, they have rules and the capability to communicate with neighboring blocks. Each block transmits a counter value to its neighboring block. In spite of their simplicity, the blocks and the robots cooperatively construct structures we designed. A remarkable aspect of this method is that the structure is stable globally, but the blocks are locally attached or detached frequently. We can say that the structure is under dynamic equilibrium. This characteristic is advantageous to adaptability to environmental changes. It is also confirmed by small simulations.

Acknowledgments This work was partially supported by a Grant-in-Aid for Scientific Research on Innovative Areas "Molecular Robotics" (No. 24104005) of The Ministry of Education, Culture, Sports, Science, and Technology, Japan.

References

1. Deneubourg, J.L., et al. The dynamics of collective sorting robot-like ants and ant-like robots. In: Animals to Animats. 356–363 (1990)
2. Beckers, R., Holland, O.E., Deneubourg, J.L.: From Local Actions To Global Tasks: Stigmergy and Collective Robotics, pp. 181–189. Artificial Life IV, MIT Press (1994)
3. Melhuish, C., Holland, O., Hoddell, S.: Collective sorting and segregation in robots with minimal sensing. In: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior, pp. 465-470 (1998)
4. Lumer, E., Faieta, B.: Exploratory database analysis via self-organization. In: Proceedings of the Computer Assisted Information Retrieval (1995)
5. Maris, M., Boeckhorst, R.: Exploiting physical constraints: heap formation through behavioral error in a group of robots. Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. **3**, 1655–1660 (1996)
6. Parker, C., Zhang, H.: Robot collective construction by blind bulldozing. IEEE Int. Conf. Syst. Man Cybern. **2**, 59–63 (2002)
7. Werfel, J.: Building blocks for multi-robot construction. In: Distributed Autonomous Robotic System 6, pp. 285–294. Springer (2007)
8. Stewart, R.L., Russell, R.A.: A distributed feedback mechanism to regulate wall construction by a robotic swarm. Adapt. Behav. **14**(1), 21–51 (2006)
9. Grasse, P.P.: La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs, Insectes Sociaux **6**, 41–81 (1959)
10. Deneubourg, J.L.: Application de l'ordre par fluctuations a la description de certaines étapes de la construction du nid chez les Termites. Insectes Sociaux **24**, 117–130 (1977)
11. Skarka, V., Deneubourg, J.L., Belic, M.R.: Mathematical model of building behavior of Apis mellifera. J. Theor. Biol. **147**, 1–16 (1990)
12. Theraulaz, G., Bonabeau, E.: Coordination in distributed building. Science **269**, 686–688 (1995)
13. Theraulaz, G., Bonabeau, E.: Modelling the collective building of complex architectures in social insects with lattice swarms. J. Theor. Biol. **177**, 381–400 (1995)
14. Werfel, J., Yaneer, B-Y., Rus, D., Nagpal, R.: Distributed construction by mobile robots with enhanced building blocks. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2787–2794 (2006)
15. Werfel, J., Petersen, K., Nagpal, R.: Distributed multi-robot algorithms for the TERMES 3D collective construction system. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (2011)
16. Werfel, J., Bar-Yam, Y., Nagpal, R.: Building patterned structures with robot swarms. In: International Joint Conference on Artificial Intelligence (2005)
17. Shimizu, M., Tsukidate, T., Sugawara, K., Ishiguro, A.: Dynamic self-assembly based on dynamic equilibrium of building and scrapping. In: Proceedings of JSME Conference on Robotics and Mechatronics, 2A1-G09(1)-(4) (2010)

Cooperative Mobile Robot Control Architecture for Lifting and Transportation of Any Shape Payload

B. Hichri, L. Adouane, J.-C. Fauroux, Y. Mezouar and I. Doroftei

Abstract This paper addresses cooperative manipulation and transportation of any payload shape, by assembling a group of simple mobile robots (denoted m-bots) into a modular poly-robot (p-bot). The focus is made in this paper on the chosen methodology to obtain sub-optimal positioning of the robots around the payload to lift it and to transport it while maintaining a geometric multi-robot formation. This appropriate positioning is obtained by combining the constraint to ensure Force Closure Grasping (FCG) for stable and safe lifting of the payload and the maximization of the Static Stability Margin (SSM) during the transport. A predefined control law is then used to track a virtual structure in which each elementary robot has to keep the desired position relative to the payload. Simulation results for an object of any shape, described by a parametric curve, are presented. Additional 3D simulation results with a multi-body dynamic software validate our proposal.

Keywords Cooperative mobile robots · Control architecture · Payload transport and co-manipulation · Force closure grasping · Static stability margin

B. Hichri (✉) · L. Adouane · J.-C. Fauroux · Y. Mezouar
Institut Pascal, Clermont Ferrand, France
e-mail: Bassem.Hichri@ifma.fr

L. Adouane
e-mail: Lounis.Adouane@univ-bpclermont.fr

J.-C. Fauroux
e-mail: Jean-Christophe.Fauroux@ifma.fr

Y. Mezouar
e-mail: Youcef.Mezouar@univ-bpclermont.fr

I. Doroftei
Gheorghe Asachi Technical University of Iasi, Iasi, Romania
e-mail: idorofte@mail.tuiasi.ro

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_13

1 Introduction

In recent years, many researches were oriented to survey and design collaborative mobile robotic systems [1, 2] gathering different engineering and science disciplines. This blend between those disciplines allows the design of autonomous systems able to interact with the environment without human mediation and also to achieve diverse complex tasks or infeasible by humans, such as exploring dangerous and/or unreachable areas [3] or navigation in formation for a group of autonomous robots [4]. Autonomous mobile robots have the ability for sensing and reacting in the environment by acquiring additional abilities. They can also collaborate when a task needs more than one robot, such as heavy objects co-manipulation or transport [3, 5–7]. The aim of our research is to co-manipulate and to transport objects using a group of mobile robots. We aim to design an innovative architecture for payload transport on structured environment. Collaborative robots behaviors may be also interesting for transporting tasks with mobile robots. Many robotic examples can be mentioned such as in [6, 8–11]. Our goal in the C^3Bots project (Collaborative Cross and Carry mobile roBots) is to design several mobile robots with a simple mechanical architecture called m-bots that will be able to autonomously co-manipulate and transport objects of any shape by connecting together. The resulting poly-robot system, called p-bot, will be able to solve the so-called removal-man-task to transport object of any shape and mass repartition. Reconfiguring the p-bot by adjusting the number of m-bots allows to manipulate heavy objects with any shape, particularly if they are wider than a single m-bot. During the manipulation, the grasping task [12, 13] is a crucial phase for payload lifting and if it fails the whole task cannot be achieved.

To ensure the co-manipulation task, he group of m-bots must succeed to ensure the payload Force Closure Grasping (FCG) [12–17] until putting it on their top platform. FCG refers to Newton laws which allows to ensure the payload immobility [13]. In the aim of ensuring object stability, which is the goal of any used grasping strategy, several methods have been developed using various approaches. Avoiding too large forces allows to reduce the power for the manipulator's actuation and the deformation of the manipulated object. A grasp is considered stable when a miniature disturbance on the position of the manipulated object or contact force, generates a restoring wrench that brings the system back to a stable configuration [12]. In [18], Nguyen presents an algorithm for stable grasps construction and he proved the possibility of making stable all 3D force closure grasps. According to [12, 13], a grasping strategy should ensure stability, task compatibility and adaptability to novel objects. Analytical and empirical approaches were developed in different literatures to ensure a stable grasping. The former approach choose the manipulator configuration and contact positions with kinematical and dynamical formulation whereas empirical approaches use learning to achieve a grasp depending on the task and on the geometry of the object. Diverse analytical methods were developed to find a force closure grasp [14, 15, 17]. The latter approach avoids the complexity of computation by attempting to mimic human strategies for grasping. Datagloves and map human hand were used by researchers for empirical approaches to learn

the different joint angles [19, 20], hand preshape [21]. Vision based approach is also used to demonstrate grasping skills. A robot can track an operator hand for several times to collect sufficient data [22, 23].

Payload stability during movement is evaluated according to developed metrics in literature. In the late sixties, stability margin metrics were developed and classified mainly in two categories: static [1, 6, 7, 9, 10, 15, 16, 18, 19, 21, 23–29] and dynamic [6, 14, 18, 22, 24, 25, 30, 31] stability margins. We consider the Static Stability Margin (SSM) since our system evolves at low speed in a structured environment. This margin was defined by McGhee and Frank [29] as follows: "static stability margin is the shortest distance from the vertical projection of the centre of gravity to any point on the boundary of the support pattern". Considering the payload lifting and transport using mobile robots, stability is also ensured by coordinating the group of transporting robots which means multi-robot control problem.

The multi-robot navigation in formation is the main research area linked to the phase of payload transportation. A multitude of control architecture to deal with this task were proposed in the literature [4, 28, 32]. A multi-robot system control can be either centralized or distributed.

The control problem is discussed to provide a suitable control strategy for this task. Formation control can be classified according to recent literature, [4, 32], into three main approaches: the behavior-based approach, the leader-follower approach and the virtual structure approach.

This paper presents an algorithm allowing to determine an optimal positioning of m-bots around a general payload in order to maximize the Static Stability Margin (SSM) and to ensure Force Closure Grasping (FCG). A centralized control will be used for its higher calculation performances to calculate different desired positions according to a payload of any shape. For targets reaching and payload transport, the groups of robots will act according to centralized control approach. A predefined control law is then used to track a virtual structure in which each elementary robot has to keep the desired position relative to the payload. This paper is organized as follow: in Sect. 2 the paradigm of C^3Bots project is introduced and the general problem is presented for co-manipulation and transport using multi-robot system; Sect. 3 will present the robots positioning according to both criteria SSM and FCG computation and the multi-robot transport strategy. Simulation results for an object of any shape, described by a parametric curve, and 3D simulations with a multi-body dynamic software are also presented. Finally Sect. 4 provides a conclusion and future works.

2 Paradigm and Problem Statement

The paradigm of C^3Bots project is to co-manipulate and transport a common payload through collaboration between several similar elementary robots (see Fig. 1). Wheeled robots were selected for their versatility on various terrains and good efficiency on regular grounds compared to legs and tracks. The C^3Bots transport strategy takes inspiration from Army Ants [9] by laying the payload on top of robot's bodies,

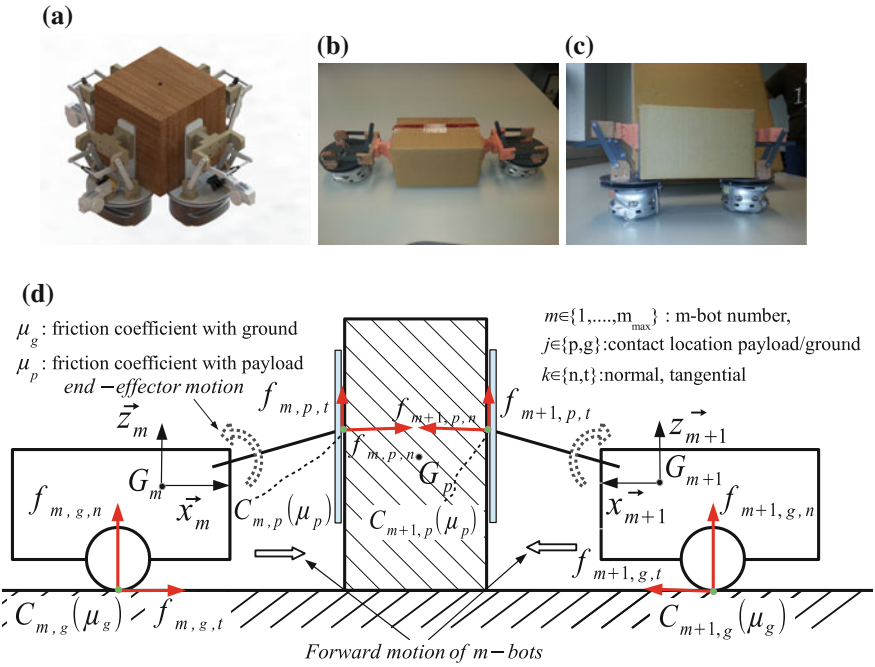


Fig. 1 Co-manipulation of a box by a group of m-bots. **a** Prototype for object lifting and transport [33]. **b** Payload prehension by two m-bots. **c** Payload lifted by two m-bots. **d** Two M-bots pushing on the payload to elevate it with parallelogram manipulator [34]

and from the structure given in [10], that has a rotative arm on top of it. The concept of modularity was also kept and each m-bot is built from two parts: a mobile platform and a manipulation mechanism [33]. The mobile platform is a single-axle Khepera robot and the manipulator is fixed on a rotary platform that lets the robot turn freely on itself when the object lays on the transporting platform. The manipulator has a parallelogram structure to bring the payload from the ground to the m-bot top platform with a circular trajectory [34].

The resulting p-bot system (cf. Fig. 1a, c) is thus allowed to translate along any direction and rotate around any point in the ground plane. Before starting the transport task, the m-bots have to achieve the co-manipulation process using the mechanism presented in [33] and detailed in [34]. Its role is to hold firmly the payload and to ensure FCG [16] to lift the object by applying a sufficient normal force $f_{m,p,n}$ (cf. Figs. 1d and 4) which generates a vertical tangential lifting force $f_{m,p,t}$ (cf. Fig. 1d) with:

$$f_{m,p,n} \in [0,\, f_{max}] = [0,\, \mu_g m_m g] \text{ and } f_{m,p,t} \in [0,\, \mu_p \mu_g m_m g] \tag{1}$$

μ_p is the payload-end-effector friction coefficient; μ_g the wheel-ground friction coefficient; m_m is the robot mass and g is the gravity. The value of f_{max} is obtained while applying the well known resultant of the force/moment for the all system (First and Second principle of Newton). We obtained thus a simple formulation of f_{max} while taking into account the mentioned parameters. To improve the system efficiency in term of payload holding and avoiding its slipping, an additional mechanism, that ensures the payload tightening and avoids friction uncertainties, is under development.

The minimum number m_{min} of m-bots that have to be used to lift and transport the payload is obtained according to Eq. (2). The payload is considered in this paper as an homogeneous body, its shape and weight are known and its center of mass is predetermined.

$$\sum_{m=1}^{m_{min}} f_{m,p,t} = M_{pl} g \tag{2}$$

3 Cooperative Mobile Robot Manipulation and Transport

The proposed overall cooperative manipulation and transport strategy, for any payload shape, by a group of m-bots is presented in Fig. 2. This figure gives the most important steps to be achieved during this cooperative task. The details of the chosen criteria for cooperative manipulation and transportation are given respectively in Sects. 3.1 and 3.2.

Step 1 (cf. Fig. 2) presents the first phase of the task and which consists on payload detection and estimation of its mass and gravity center position. Step 2 consists on determining the minimum number of m-bots (m_{min}) that could be used to ensure the payload lifting and transport with relative to (2). Step 3 presents the main contribution of this paper. It is detailed by the flowchart in the right side of Fig. 2 and will be discussed in Sects. 3.1.1 and 3.1.2. Sasaki et al. in [27] treated a similar problem for optimal robots positioning taking into account two criterion: the payload stability and the energy consumption. It was considered that the positioning is optimal when the payload is stable and the robots consume the minimum of energy (according to the data received from the robots sensors). In the proposed strategy, the m-bots positioning is optimal when FCG and SSM are ensured. Finally, Step 4 corresponds to multi-robot transport the payload toward the assigned final pose, this part will be detailed in Sect. 3.2.

3.1 Cooperative m-bots Positioning and Co-manipulation

Since the features of the payload are known (step 1 in Fig. 2) the minimum number of m-bots (m_{min}) is obtained while using Eq. 2 (step 2), the group of m-bots must be well positioned around the payload (step 3) to permit to safely lift it and to maintain

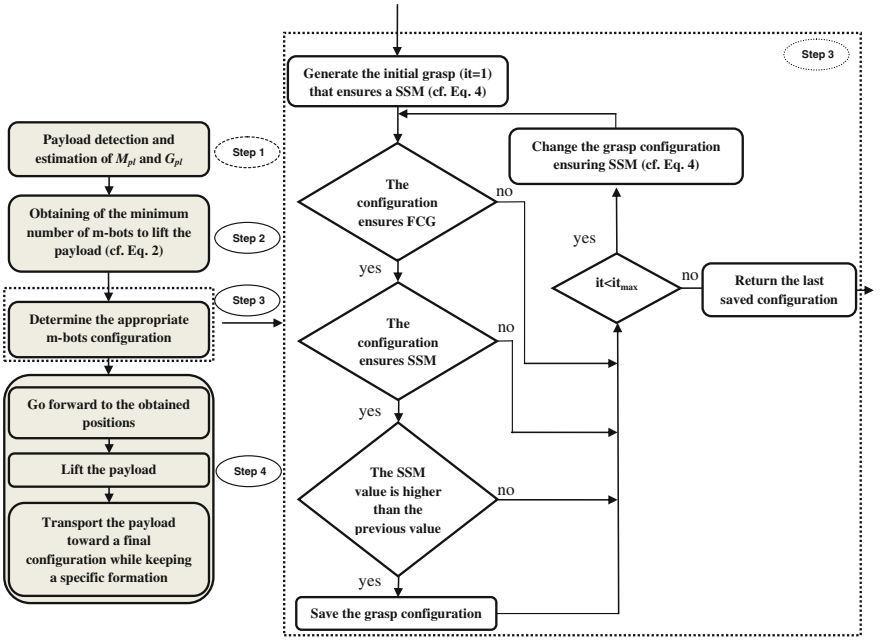


Fig. 2 Flowchart given the sequenced steps for the co-manipulation and transportation of any payload shape

a well stability of the payload in the top of the p-bot during the transportation phase (step 4). During this manipulation phase (sub-step 2 in step 4), FCG (cf. Sect. 3.1.1) as well as SSM (cf. Sect. 3.1.2) must be thus ensured to lift and transport safely the object (cf. details given for Step 3 in Fig. 2).

3.1.1 Force Closure Grasping

Force closure grasping problem is extensively treated and studied for objects manipulation using multi fingered robotic hand [35, 36]. This problem was adapted to mobile robot co-manipulation and transport in C^{3}Bots project to ensure lifting and transport task.

The co-manipulation problem (cf. Sect. 2) is restricted to a 2D problem in plane $(O, \mathbf{x}, \mathbf{y})$ while robots are acting simultaneously and applying a tightening forces on the payload on the same plane (Fig. 3).

The aim of this part is to ensure force closure grasping when choosing the m-bots positions which returns to fully constraint the payload motion with m_{min} m-bots. In other words, the static equilibrium must be ensured while positioning the group of mobile robots. The problem of force closure grasping is studied under the following assumptions (cf. Fig. 3c):

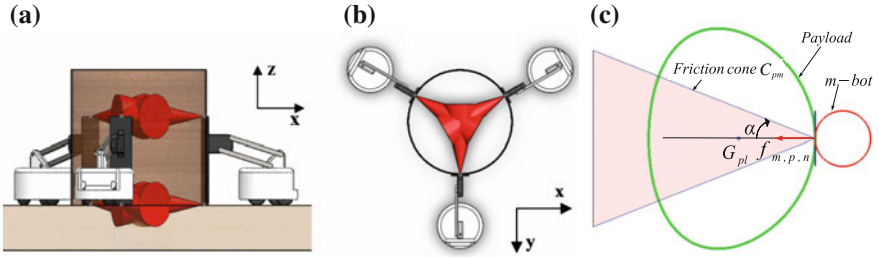


Fig. 3 Applied tightening forces on the payload. **a** *Side view*. **b** *Top view*. **c** m-bot planar contact

- A contact force lies inside the friction cone centred about the normal direction to the contact surface with half angle α.
- The tangent of α represent the friction coefficient.
- The friction cone of the mth contact is denoted C_{pm}.

A necessary and sufficient condition to have force closure is that the intersection of three friction cones is not empty [17]. This condition was extended to m_{min} m-bots. In [17], the treated problem concerns multi fingered hand grasping although the problem treated in this paper focuses on co-manipulation using a group of modular mobile robots. The proposed algorithm is based on ensuring force closure if forces and moments equilibrium satisfy (3) and when the payload center of mass is inside the friction cones intersection (4). The later condition allows to reduce the moments generated on the payload by the m-bots because the direction of the applied force on the plane is closer to the gravity center.

$$\sum_{m=1}^{m_{min}} (P_m G_{pl} \otimes f_{m,p,n}) = 0; \sum_{m=1}^{m_{min}} f_{m,p,n} = 0 \tag{3}$$

$$G_{pl} \in Convexhull(\cap C_{pm}) \mid m = 1..m_{min} \tag{4}$$

where C_{pm} presents the friction cone for the contact force on P_m and $f_{m,p,n}$ is the applied normal on the payload (cf. Fig. 3c).

3.1.2 Static Stability Margin (SSM)

In this part, Static Stability Margin (SSM) is considered to ensure the payload stability during the transporting phase. Stability margins were extensively studied for walking mobile robots [26, 31, 37]. In C^{3}Bots project, to ensure a stable payload transport, the Static Stability Margin (SSM) is a crucial criterion for a successful task achievement. Before describing the proposed algorithm for m-bots positioning ensuring an optimal SSM during object transport using m-bots, let's detail the following assumptions (cf. Fig. 4):

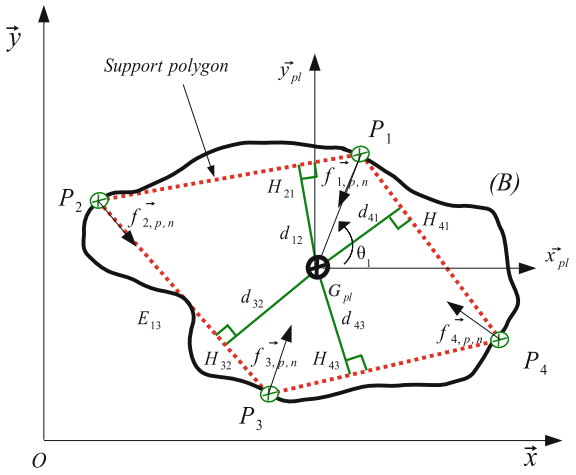


Fig. 4 Support polygon formed by four robots positioned at $P_{m|m=1..4}$

- The payload shape from the top view is a closed curve (B) and defined by polar curve defined by $P(\theta)$; $\theta \in [0, 2\pi]$.
- In function of the payload mass M_p, m_{min} is the minimum number of m-bots allowing to lift and transport the object.
- The payload center of mass is denoted G_{pl}.

Let $R(G_{pl}, \mathbf{x}_{pl}, \mathbf{y}_{pl}, \mathbf{z}_{pl})$ be the frame linked to the payload with respect to the reference frame $R(O, \mathbf{x}, \mathbf{y}, \mathbf{z})$ (cf. Fig. 4). Cartesian coordinates will be used in the proposed algorithm. As given in Sect. 2, $P(\theta)$ be the parametric description of the payload closed boundary (B). $P_{m|m=1..m_{min}}$ are the m-bots positions, $H_{m,m+1}$ is the projection of the payload center of mass G on the edge linking two consecutive points P_m and P_{m+1} and $d_{m,m+1}$ is the stability margin on the same edge. P_m and $P_{m_{min}+1}$ are confounded and as a consequence $d_{m,m_{min}+1}$ is equal to $d_{m_{min},1}$.

The idea behind the algorithm is to run through (B) and to find the set of points P_m ensuring a maximal SSM while maximizing the objective function (5). The constraint imposed by (6) must be satisfied for m_{min} m-bots ≥ 3 which gives a necessary condition to keep the center of mass G_{pl} inside the polygon $(P_1..P_m)$

$$f(\theta_m, \ldots \theta_{m_{min}}) = \text{Min}(d_{m,m+1}) \mid m = 1..m_{min} \tag{5}$$

$$\theta_{m+1} - \theta_m < \pi \mid m = \{1...m_{min}\} \tag{6}$$

In the case where we have only two m-bots to co-manipulate the object, the constraint expressed by (6) is not considered and the robots are positioned in opposed positions which means $\theta_{m+1} - \theta_m = \pi$. For each configuration where n m-bots ≥ 3, the algorithm aims at determining the equation of the line $P_m P_{m+1}$ and at computing the shortest distance of $G_{pl}(x_{G_{pl}}, y_{G_{pl}})$ from it.

Then $d_{m,m+1}$ is calculated by (7) which represent the stability margin relative to each edge and the static stability margin SSM given by (5). P_m coordinates are expressed in $R(G_{pl}, \mathbf{x}_{pl}, \mathbf{y}_{pl}, \mathbf{z}_{pl})$ (cf. Fig. 4).

$$d_{m,m+1} = d(G, (P_m P_{m+1})) = \frac{x_G \frac{y_{P_{m+1}} - y_{P_m}}{x_{P_{m+1}} - x_{P_m}} - y_G + y_{P_m} - x_{P_m} \frac{y_{P_{m+1}} - y_{P_m}}{x_{P_{m+1}} - x_{P_m}}}{\sqrt{(\frac{y_{P_{m+1}} - y_{P_m}}{x_{P_{m+1}} - x_{P_m}})^2 + 1}} \quad (7)$$

3.1.3 Simulation Results

The proposed algorithm allows to determine a sub-optimal configuration for a group of mobile robots in order to lift and transport a payload of any shape. Two criteria have been respected (FCG and SSM) which reduces the total configurations to be tested by the algorithm taking into consideration (3) and (4). The Algorithm was simulated by using an Intel Core i5 2400 CPU 3.1 GHz system. Figure 5 presents the simulation results for the developed algorithm for robots positioning in order to guarantee an optimal static stability margin respecting the force closure condition. The blue bold polygon presents the polygon of support ensuring the optimal SSM (cf. Sect. 3.1.2), the thin blue lines presents the friction cones sides and the intersection is presented by contrasted area resulted by the superposition of friction cones. It is shown how the algorithm keeps the payload center of mass G_{pl} inside the intersection area and it allows to build a polygon of support ensuring the payload stability during the transport. The duration to find results depends on the chosen steps of θ_m to run throw the payload curve.

The payload stability during the lifting phase was simulated with respect to both criteria (SSM and FCG) using ADAMS multi-body dynamic software to validate the proposed algorithm (cf. Fig. 2) while testing the m-bots performances when they are positioned to co-manipulate the object. Figure 6 shows that the robots ensure the payload lifting without loss of stability of the lift. Videos for simulation are visible under [38].

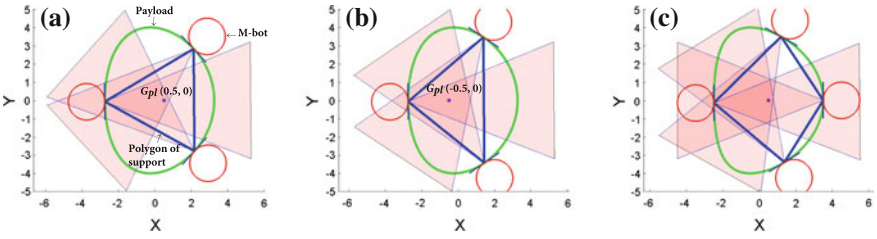


Fig. 5 Simulation results. **a**–**b** 3 m-bots positioning with different configuration according to the localization of the payload center of mass; **c** 4 m-bots positioning

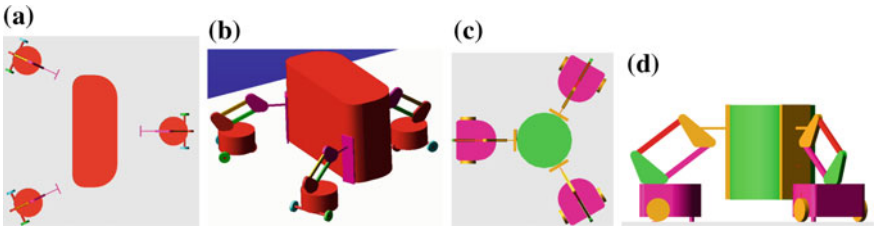


Fig. 6 Multibody simulation results with ADMAS software. *Top view* (**a**, **c**), and 3D lifting phase (**b**, **d**)

3.2 Multi-robot Transport

After lifting the payload, which is positioned now on the top of the p-bot, the group of m-bots must transport the payload toward a final configuration. During this last phase (Step 4 in Fig. 2), and in order to guarantee the payload stability, the p-bot should navigate as rigid formation shape and for this, a virtual structure architecture was used [4]. After the end of Step 3, each m-bot receives its attributed position which insures the sub-optimal p-bot positioning that permits to ensure Force Closure Grasping (FCG) and to maximize the Static Stability Margin (SSM) during the transport. For transport task, the m-bots have to reach their goals, computed using the algorithm presented in the previous section (cf. Step 4 in Fig. 2). After reaching the desired positions, the transport task starts considering that the payload lays on robots bodies. To avoid payload slippage, the group of m-bots has to track a fixed position relative to the object when it follows a trajectory. In this section, a control law is proposed to solve the goal reaching problem (P_m in Sect. 3.1.2) and the navigation as Virtual Structure (VS) of the set of m-bots. In VS approach [4, 32], the entire formation is considered as a rigid body and the notion of hierarchy do not exist. The control law for each entity is derived by defining the VS dynamics and then translate the motion of the VS into the desired motion of each elementary robot. The main advantages of this approach are its simplicity to prescribe the coordinate behavior of the group and the maintaining of the formation during manoeuvres.

The result of the algorithm for a given object shape described by a parametric curve (B) is a set of n targets to be reached by the m-bots. Considering a unicycle mobile robot, the state vector $X_m = (x_m, y_m, \theta_m)^T$ denotes the position of the m^{th} robot center of mass $G_m(x_m, y_m)$ and its orientation θ_m with respect to **x** axis of the global frame. The m-bots control inputs are the forward velocity V and the angular velocity ω.

Let e be the error between the m-bot current pose and the desired pose defined by $X_{dm} = (x_{dm}, y_{dm}, \theta_{dm})^T$: $e = X_{dm} - X_m$.

After positioning the m-bots, they must keep their desired position (x_{dm}, y_{dm}) with respect to the payload center of mass G_{pl} and must respect the following conditions during the task achievement:

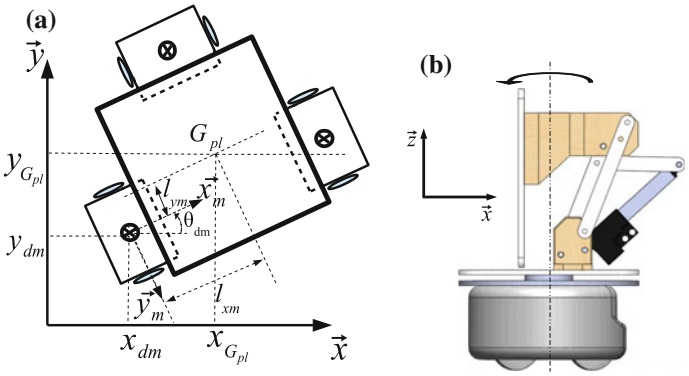


Fig. 7 M-bot position and mobility during payload transport. **a** Desired position of the robot relative to the payload; **b** free steering of the mobile platform relative to the manipulator

$$\begin{aligned} x_{dm} &= x_{G_{pl}} + l_{xm}\cos\theta_{dm} - l_{ym}\sin\theta_{dm} \\ y_{dm} &= y_{G_{pl}} + l_{xm}\sin\theta_{dm} + l_{ym}\cos\theta_{dm} \end{aligned} \tag{8}$$

where l_{xm} and l_{ym} (cf. Fig. 7a) are the relative distances $G_m G_{pl}$ according the axis $\mathbf{x}_m$ and $\mathbf{y}_m$ respectively. These two distances define rigid links maintaining the robot position with respect to G_{pl}. It is to be noted that the mobile platform has a steering mobility around its vertical axis z (cf. Fig. 7b). This mobility allows to each robot to rotate around itself ($V_m = 0$ and $\omega_m =$ Constant (cf. Eq. (9))) while maintaining the payload static on its top. According to this effector new degree of freedom, the group of mobile robots could ensure easily the payload approach, lifting and transportation.

The used control law [4] is given by (9):

$$\begin{aligned} V_m &= V_{max} - (V_{max} - V_d)e^{-(d_m^2/\sigma^2)} \\ \omega_m &= \omega_{Sm} + k\theta_m \end{aligned} \tag{9}$$

- V_m and ω_m are the linear and angular velocities of the m-bot.
- V_{max} is the maximum linear speed of the m-bot.
- V_d is the desired velocity of the p-bot and considered to be constant.
- $d_m = \sqrt{e_x^2 + e_y^2}$ is the current distance between the mth robot and its desired target.
- ω_{Sm} is the angular velocity of set point angle θ_{Sm} applied to the robot in order to reach the desired goal: $\omega_{Sm} = \dot{\theta}_{Sm}$
- σ, k are positive constants (control law gains).

The control law was simulated for a group of three m-bots transporting an object. Figure 8a presents the goal reaching problem with k = 22 and $\sigma = 0.1$. In order that the m-bots reach the desired positions, the desired speed when reaching the goal is set to zero and then the whole structure will navigate with a speed of 10 cm/s (Fig. 8d, e and f). The payload lays on robot bodies during transport and the group

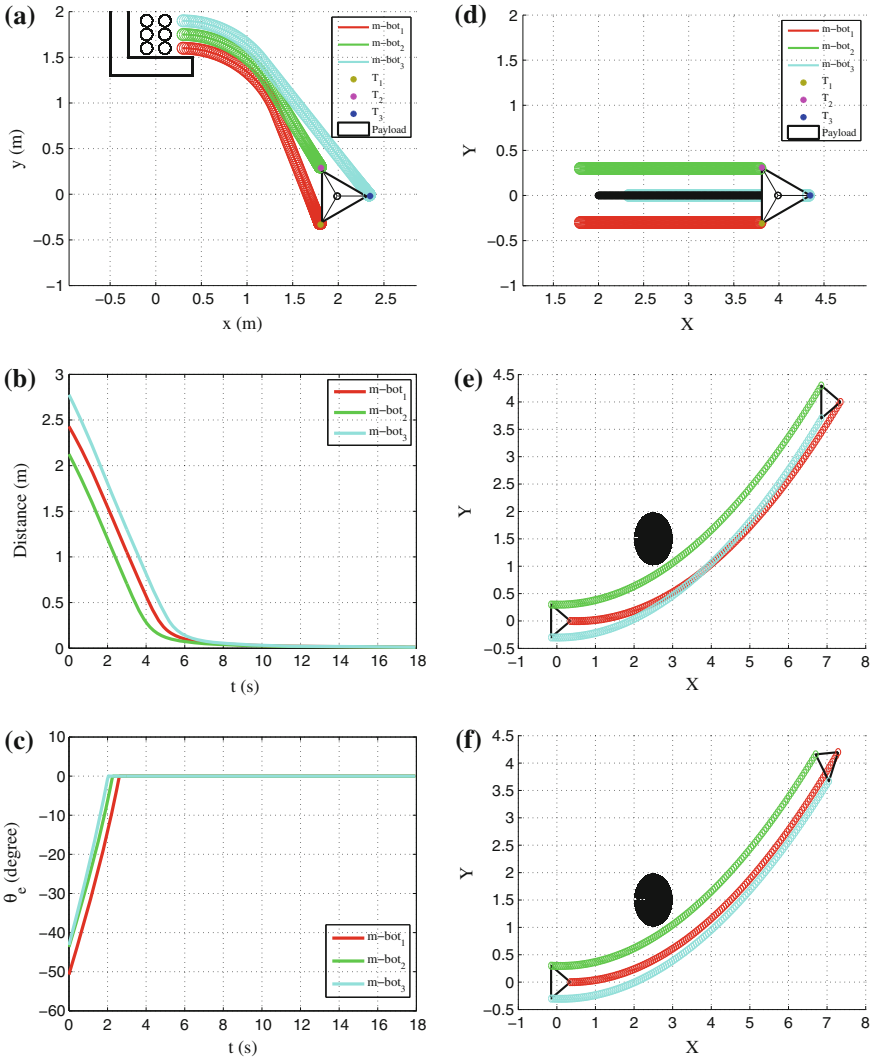


Fig. 8 M-bots target reaching (TR) and Virtual structure (VS) navigation: **a** Trajectories of the m-bots reaching the desired positions; **b** position error for TR; **c** angle Error for TR; **d** the p-bot is navigating as a rigid Virtual Structure (VS); **e** the p-bot avoids the obstacle and keeps the same orientation; **f** the p-bot avoids obstacle and changes the payload orientation

of m-bots is navigating while maintaining constant distances. Figure 8b shows the convergence of the position error e to zero during target reaching phase. Figure 8c presents the angular error for each robot. One can note the convergence to zero of the error which shows the target reaching achievement. Figure 8d, e and f) show respectively the payload transport in a straight line, considering obstacle avoidance

while keeping the payload orientation and finally with a new payload orientation. One can note that all m-bots keep a null position errors which means that the formation is properly maintained and that slippage avoidance and task performance are ensured. It is important to notice, that in this paper, we suppose a centralized control of the fleet of robots, thus, the movement of the virtual structure is already defined according to the configuration of the environment. Indeed, the focus of this paper is on the presentation of the virtual structure and the way how each elementary robot keeps the desired position relative to the payload.

4 Conclusions and Future Work

This work takes place within the C^{3}Bots project, that aims to design simple robot entities (m-bots) able to co-manipulate and transport payloads of any shape by aggregating in a modular way into a poly-robot (p-bot). This work has the ambition to combine two criteria in an original way:

- On one side, the Static Stability Margin (SSM), generally used for legged locomotion.
- On the other side, Force Closure Grasping (FCG), used for stable multi-finger manipulation.

The m-bots used in this work include in their lower part a wheeled-axle, which is similar to a foot of a multi-leg mobile robot, and in their top part a manipulator acting like the finger of a robotic hand. The resulting p-bot ensures the stable payload grasping and transport. An algorithm was developed in order to search the optimal positions of n unicycle m-bots that ensure force closure grasping and maximize the static stability margin for the transport of a payload of any shape, defined by its closed curve boundary. Simulation results using a multi-body dynamic software validates our proposal and shows the ability of robots to maintain the payload stability during lifting process. A flexible control architecture was used to validate the target reaching problem while maintaining the chosen formation. This navigation was considered in a flat structured environment. Future works will consider the problem of payload manipulation and lifting in all terrain. Unreachable areas on the payload boundary will also have to be taken into consideration (as for example one side of a square object opposed to a wall).

Acknowledgments The C^{3}Bots project acknowledges the following entities: LABEX IMobS3 Innovative Mobility: Smart and Sustainable Solutions, the French National Centre for Scientific Research (CNRS), Auvergne Regional Council and the European funds of regional development (FEDER).

References

1. Souma Alhaj Ali, M.G.: Mobile Robotics, Moving Intelligence (2006)
2. Siegwart, R., Nourbakhsh, I.R.: Introduction to Autonomous Mobile Robots. The MIT Press (2004)
3. Wilcox, B.H., Litwin, T., Biesiadecki, J., Matthews, J., Heverly, M., Morrison, J., Townsend, J., Ahmad, N., Sirota, A., Cooper, B.: Athlete: a cargo handling and manipulation robot for the moon. J. Field Robot. **24**(5), 421–434 (2007)
4. Benzerrouk, A., Adouane, L., Lequievre, L., Martinet, P.: Navigation of multi-robot formation in unstructured environment using dynamical virtual structures. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5589–5594 (2010)
5. Ijspeert, A.J., Martinoli, A., Billard, A., Gambardella, L.M.: Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment. Auton. Robots **11**(2), 149–171 (2001)
6. Dorigo, D.F.M.: Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robotics & Automation Magazine. In Press (2012)
7. Adouane, L., Le Fort-Piat, N.: Hybrid behavioral control architecture for the cooperation of minimalist mobile robots. In: 2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04, vol. 4, pp. 3735–3740 (2004)
8. Yamashita, A., et al.: Cooperative manipulation of objects by multiple mobile robots with tools. In: Proceedings of the 4th Japan-France/2nd Asia-Europe Congress on Mechatronics, pp. 310–315 (1998)
9. Bay, J.S.: Design of the army-ant cooperative lifting robot. IEEE Robot. & Autom. Mag. **1**, 36–43 (1995)
10. Abou-Samah, M.: Optimal configuration selection for a cooperating system of mobile manipulators. In: 2002 ASME Design Engineering Technical Conferences (2002)
11. Kernbach, S., et al.: Symbiotic robot organisms: REPLICATOR and SYMBRION projects. In: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems, pp. 62–69. New York (2008)
12. Sahbani, A., El-khoury, S., Bidaud, P.: An overview of 3D object grasp synthesis algorithms. Robot. Auton. Syst. **60**(3), 326–336 (2012)
13. El-Khoury, S., Sahbani, A., Bidaud, P.: 3D objects grasps synthesis: a survey. In: 13th World Congress in Mechanism and Machine Science. Guanajuato (2011)
14. Ding, D., et al.: Computing 3-D optimal form-closure grasps. In: IEEE International Conference on Robotics and Automation. Proceedings. ICRA, vol. 4, pp. 3573–3578 (2000)
15. Li, J.-W., Jin, M.-H., Liu, H.: A new algorithm for three-finger force-closure grasp of polygonal objects. In: IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA, vol. 2, pp. 1800–1804 (2003)
16. Roa, M.A., Suarez, R.: Finding locally optimum force-closure grasps. Robot. Comput. Integr. Manuf. **25**(3), 536–544 (2009)
17. Liu, Y.H.: Qualitative test and force optimization of 3-D frictional form closure grasps using linear programming. IEEE Trans. Robot. Autom. **15**(1) (1999)
18. Nguyen, D.: Constructing stable grasps in 3D. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 234–239 (1987)
19. Fischer, M., van der Smagt, P., Hirzinger, G.: Learning techniques in a dataglove based telemanipulation system for the DLR hand. In: 1998 IEEE International Conference on Robotics and Automation, 1998. Proceedings, vol. 2, pp. 1603–1608 (1998)
20. Ekvall, S., Kragic, D.: Interactive grasp learning based on humain demonstration. In: IEEE/RSJ International Conference on Robotics and Automation. New Orleans (2004)
21. Kyota, F., et al.: Detection and evaluation of grasping positions for autonomous agents. In: International Conference on Cyberworlds, 2005, pp. 453–460 (2005)
22. Aarno, D., et al.: Early reactive grasping with second order 3D feature relations. In: Lee, S., Suh, I.H., Kim, M.S. (eds.) Recent Progress in Robotics: Viable Robotic Service to Human, pp. 91–105. Springer, Berlin (2008)

23. Hueser, M., Baier, T., Zhang, J.: Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA, pp. 2795–2800 (2006)
24. Orin, D.E., Mcghee, R.B., Jaswa, V.C.: Interactive compute-control of a six-legged robot vehicle with optimization of stability, terrain adaptibility and energy. In: 1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes, vol. 15, pp. 382–391 (1976)
25. Papadopoulos, E.G., Rey, D.A.: A new measure of tipover stability margin for mobile manipulators. In: 1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings, vol. 4, pp. 3111–3116 (1996)
26. Estremera, J., Cobano, J.A., Gonzalez de Santos, P.: Continuous free-crab gaits for hexapod robots on a natural terrain with forbidden zones: an application to humanitarian demining. Robot. Auton. Syst. **58**(5), 700–711 (2010)
27. Sasaki, J., Ota, J., Yoshida, E., Kurabayashi, D., Arai, T.: Cooperating grasping of a large object by multiple mobile robots. In: 1995 IEEE International Conference on Robotics and Automation, 1995. Proceedings, vol. 1, pp. 1205–1210 (1995)
28. Adouane, L.: Architectures de controle comportementales et reactives pour la cooperation d'un groupe de robots mobiles. Université de Franche-Comté, PhD Thesis Report (2005)
29. McGhee, R.B., Frank, A.A.: On the stability properties of quadruped creeping gaits. Math. Biosci. **3**, 331–351 (1968)
30. Grand, C., et al.: Stability and traction optimization of a reconfigurable wheel-legged robot. Int. J. Robot. Res. **23**(10–11), 1041–1058 (2004)
31. Queiroz, C.: A study on static gaits for a four legged robot. In: International Conference CONTROL'2000. Cambridge, UK (2000)
32. Van den Broek, T.H.A., et al.: Formation control of unicycle mobile robots: a virtual structure approach. In: Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 2009 28th Chinese Control Conference. CDC/CCC, pp. 8328–8333 (2009)
33. Hichri, B., Fauroux, J.C., Adouane, L., Mezouar, Y., Doroftei, I.: Design of collaborative, cross and carry mobile robots (C3Bots). Adv. Mater. Res. **837**, 588–593 (2013)
34. Hichri, B., Fauroux, J.C., Adouane, L., Doroftei, I., Mezouar, Y.: Lifting mechanism for payload transport by collaborative mobile robots. In: Flores, P., Viadero, F. (eds.) New Trends in Mechanism and Machine Science, pp. 157–165. Springer (2015)
35. Yoshikawa, T.: Multifingered robot hands: control for grasping and manipulation. Annu. Rev. Control **34**(2), 199–208 (2010)
36. Zheng, Y., Qian, W.-H.: Limiting and minimizing the contact forces in multifingered grasping. Mech. Mach. Theory **41**(10), 1243–1257 (2006)
37. Wang, Z., Ding, X., Rovetta, A., Giusti, A.: Mobility analysis of the typical gait of a radial symmetrical six-legged robot. Mechatronics **21**(7), 1133–1146 (2011)
38. Simulation results: https://www.dropbox.com/sh/d6plmdqmnizm8j6/AABy52fbl65lhC870ZBdjdQfa

A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration

Anshul Kanakia, John Klingner and Nikolaus Correll

Abstract We present a multi agent collaboration algorithm to recruit an approximate number of individually simple robots with controllable variance. We propose a sigmoid response threshold function motivated by task allocation in social insects, and describe macro-level models backed by micro-level simulations to predict the resulting team sizes and their variance. These results are further validated through physical experiments using the "Droplet" swarm robotics platform. We show that the slope of the response threshold function can be used to control the variance of group size, allowing agents to trade off deterministic team size with coordination speed, and making the proposed mechanism applicable to a variety of applications.

Keywords Swarm robotics · Multi-agent systems · Collaboration · Task-allocation · Response-threshold

1 Introduction

We propose a probabilistic, threshold based multi agent collaboration algorithm and analysis to recruit an *approximate* number of robots for a collective task. Recruiting robots to collaboratively solve a task is a canonical problem in robotics [13]. Formally introduced in the stick-pulling experiment, as described by Martinoli et al. [19, 20], to recruit exactly two robots to collaboratively pull a stick out of the ground, Lerman et al. [18] have extended this model to larger teams of constant size. In each case the result of collaboration is binary, success or failure, based on whether an *exact* number of robots are present at the collaboration site or not.

A. Kanakia (✉) · J. Klingner · N. Correll
Department of Computer Science, University of Colorado, Boulder, USA
e-mail: anshul.kanakia@colorado.edu

J. Klingner
e-mail: john.klingner@colorado.edu

N. Correll
e-mail: nikolaus.correll@colorado.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_14

We wish to extend the stick pulling model of constant group sizes of robots to include a more general case of collaboration tasks that involve only approximate robot group sizes for successful collaboration. More specifically, we deal with tasks that have the property of "concurrent benefit" where single agents must wait for a group—with a range of permissible size—to form at their collaboration site before being able to collectively begin the task and complete it successfully. Examples of such tasks include fire containment, collective transport [22], pattern recognition [3], real-time mapping of oil spills [2], determining coverage area of forest fires [17], and many others that require a subset of a swarm to coalesce and tackle a task collaboratively.

A concrete example is fire containment (See Fig. 1). Dropping incremental amounts of water on a fire will be futile until a critical mass of robots drop their water at the same time. Waiting for an exact number of robots, however, is not necessary either, motivating a task allocation scheme that will result in an average number of robots with predictable variance.

It is important to note that we do not study any particular task in detail but rather, outline a general approach to modeling scenarios with the aforementioned properties. Here, tuning the permissible variance allows one to tune the likelihood with which collaborations happen. Formally understanding the dynamics of the underlying coordination mechanisms allows the designer not only to predict performance, but also to optimize a swarming system [8].

The proposed algorithm and model is inspired from task allocation in biological systems such as ant colonies [5, 16]. We employ the use of a response threshold sigmoid function that probabilistically triggers the beginning of a collaboration step between robots at the same collaboration site. We study this approach via macroscopic models and microscopic simulations. The sigmoid function used in our model is commonly referred to as the Logistic function and has control parameters that allow us to alter its offset and slope. We study the effects that changing these parameters has on the system-wide behavior of the robot swarm. We also draw comparisons between this collaboration model and similar models used by Lerman et al. for the n-robot stick pulling experiment [18] and discuss situations where it is beneficial to use one model over the other.

The rest of the paper is organized as follows. Section 2 introduces the task allocation algorithm that we will be studying throughout the course of the paper. Section 3 provides a mathematical basis for the threshold based collaboration model and attempts to explain how the voting strategy being employed affects group sizes and their variance. Section 4 provides an explanation of the agent level controller in the system. This section further outlines the experimental setup being used to run simulations and the important parameters of the system. We compare collaboration rates of our model with the constant group size model introduced by Lerman et al. [18] using micro-level Gillespie simulation in Sect. 4, showing that the dynamics are comparable for similar team sizes, yet allow us to tune the variance of the resulting group size. We also discuss results obtained from conducting real physical experiments and compare them to microscopic simulation results. Finally, Sects. 6 and 7 provide

discussion of the drawbacks and limitations of the proposed model and scenarios where it could fail as well as discussing possible avenues for future work.

1.1 Related Work

Task allocation is a canonical problem in multi-robot systems [13]. Whereas capable robots might be able to approximate optimal task allocation, e.g., using market-based approaches [1, 23] or using leader-follower coalition algorithms [7], probabilistic algorithms are of particular interest for swarm robotics with individually simple controllers [10]. Recruitment of an exact number of robots to a particular task has been extensively studied using the "Stick Pulling" experiment [18, 19]. The problem of distributing a swarm of robots across a discrete number of sites/tasks with a specific desired distribution has been studied in [4, 8]. The proposed algorithm extends upon the first group of work, and we show how the proposed stochastic task allocation algorithm reduces to the ones described in [18, 19] when using appropriate parameters. Mather [21] instead presents a stochastic approach that is a hybrid between the work in [4] and [19], allowing allocation to tasks requiring a varying number of robots. While a response threshold function can also be applied to the swarm distribution problem, this problem is complementary to the problem of recruiting teams of varying sizes addressed in this paper.

2 Response Threshold-Based Task Allocation

We consider a generic collaboration task with m uniformly distributed collaboration sites within a flat arena with area A. A swarm of individually simple robots such as the *Droplet* platform [11, 15] is deployed within the arena, uniformly and at random. The number of robots being used per experiment varies, as we discuss results for a number of different scenarios. Collaboration sites in the arena can be of various sizes and configurations.

Each individual agent is capable of locomotion [15] and local sensing [11]. The agents do not require global positioning and no centralized controller exists, but we assume each agent to be capable of local omnidirectional communication with other agents within its communication range. The agents are also capable of sensing the boundary of a collaboration site—we assume that sites have easily distinguishable boundary regions, as shown in Fig. 1, for the purposes of the model studied in this paper.

The objective of each agent in the robot swarm is to find a collaboration site in the arena and perform a collective task with other agents at that site. The precise details of the collective task are not important for the purpose of understanding the coordination mechanism. We assume the actual collective task takes each agent a probabilistic, finite amount of time to complete. Once collaboration is complete, the

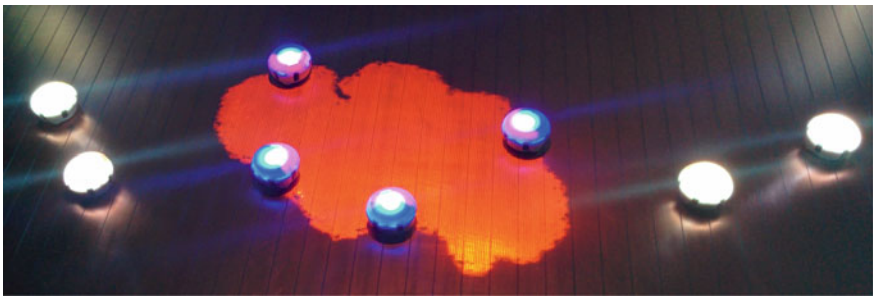


Fig. 1 A visual representation of the collaboration experiment firefighting scenario using the *Droplet* swarm robotic platform

agent detaches itself from its current site and returns to searching for other sites in the arena.

It is perfectly reasonable to assume that agents arrive at the same collaboration site after having just completed a task there (possibly unsuccessfully) but will now be part of a new collaboration group. Each agent individually decides whether or not to collaborate at a given time step, while waiting at a collaboration site. If the majority of agents at that site decide to collaborate then the entire population is recruited for the task and thus a collective consensus is reached using a majority voting scheme. Here, we consider a "majority" to mean exactly half or more of a given population.

An individual agent-i's willingness to collaborate is a stochastic term governed by a sigmoid based threshold function that takes as input, the number of agents $x_{\hat{m}}$ currently at the same collaboration site as agent-i and outputs a probability of collaboration using control parameters θ and τ:

$$\mathscr{S}(x_{\hat{m}}) = \frac{1}{1 + e^{\theta(\tau - x_{\hat{m}})}} \tag{1}$$

The parameter θ controls the slope of the sigmoid function, while τ controls its offset along the x axis, as seen in Fig. 2. Each agent is independently responsible for estimating the group size $x_{\hat{m}}$ at a given time either by direct sensing or by communication. In practice, this involves building a list of unique identifiers of the agents sharing its collaboration site. The overall algorithm, followed by each individual agent in the system, is provided in Algorithm 1.

Note that the proposed response threshold function is different from [5], who uses high-order polynomials. While these functions work well in regimes with moderate slope, they create numerical problems when approximating unit-step-like responses such as those (implicitly) used in [18]. We particularly chose the Logistic function from the large class of sigmoid functions due to the intuitive nature of the parameters τ and θ.

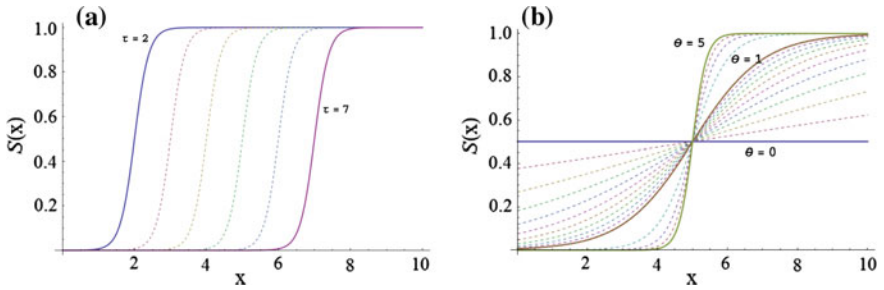


Fig. 2 Sigmoidal response threshold function and its parameters. **a** Changing τ offsets the curve along the x axis, allowing to set the desired mean team size. **b** Changing θ changes the slope at the point $x^* = \tau$, $\mathscr{S}(x^*) = 0.5$, allowing to control the team's variance

Algorithm 1 Task allocation algorithm for an individual agent using the sigmoid threshold function

```
function TASK_ALLOCATION(θ, τ)
    estimate ← discover_group_size()
    decision ← run_sigmoid(estimate, θ, τ)
    communicate_decision(decision)
    decisions[] ← gather_decisions()
    result ← Count(decisions[], true)        ▷ Count() Returns the number of successes in the decisions
    if result ≥ (estimate/2) then
        Collaborate()
        return
    else
        Task_Allocation(θ, τ)
    end if
end function
```

3 Macroscopic Analysis

In this section we study how the local parameters τ and θ from an individual agent's sigmoid threshold function affect formation of groups of different sizes at the macroscopic system level.

3.1 A General Model of Probabilistic Task Allocation

Equation (1) is a cumulative probability density function approaching 1.0 as the number of agents approaches infinity, that is $\lim_{x\to\infty} \mathscr{S}(x) = 1$. For $\theta \to \infty$, Eq. (1) approximates the unit step:

$$\lim_{\theta\to\infty} \frac{1}{1+e^{\theta(\tau-x_{\hat{m}})}} \approx \begin{cases} 1 & x_{\hat{m}} > \tau \\ 1/2 & x_{\hat{m}} = \tau \\ 0 & x_{\hat{m}} < \tau \end{cases} \tag{2}$$

Although, unlike the unit step function, the limit on the left in (2) is always continuous, even at $x_{\hat{m}} = \tau$ where the value of the sigmoid is $1/2$. The proposed model is therefore a generalization of the "stick-pulling" task allocation model with deterministic team size [18], allowing us to tune the variable resulting group sizes using the tuning parameters τ and θ in (1).

3.2 From Individual Choices to Team-Level Collaboration

Assuming the agents to be loosely synchronized, e.g., by considering decisions within a finite window of time, determining a majority vote corresponds to a Bernoulli trial with each agent flipping a biased coin—the bias being computed using the sigmoid function—to decide whether or not to collaborate in the next time step. The probability that exactly k agents collaborate from a population of n agents at a collaboration site is given by the probability mass function (PMF) of a Binomial distribution.

$$B(n,k) = \binom{n}{k} \mathscr{S}(n)^k \left(1-\mathscr{S}(n)\right)^{n-k} \tag{3}$$

Since we care about the case when half or more of the agents $(n/2)$ decide to collaborate, the probability $P(n)$ that half or more agents in a group of n collaborate is the cumulative probability of the above PMF from $k = n/2$ to $k = n$.

$$P(n) = \sum_{i=n/2}^{n} \binom{n}{i} \mathscr{S}(n)^i \left(1-\mathscr{S}(n)\right)^{n-i} \tag{4}$$

This equation describes the probability with which a group of size n at a given collaboration site will decide to successfully collaborate. Note that (4) is only an approximation for odd n, which requires rounding $\lceil n/2 \rceil$ to the next integer.

For large group sizes, the Binomial distribution approximates the Normal distribution and (4) reduces to

$$P(n) = \int_{n/2}^{n} \mathscr{N}\left(n\mathscr{S}(n), n\mathscr{S}(n)(1-\mathscr{S}(n))\right) \tag{5}$$

Therefore, in a group of size n, and n reasonably high (see below), an average of $n\mathscr{S}(n)$ robots will collaborate with group sizes of variance $n\mathscr{S}(n)(1-\mathscr{S}(n))$. In the special case of $n = \tau$, i.e., the group size has the desired value of τ, (5) evaluates to $P(\tau) = \mathscr{S}(\tau) = 0.5$. Therefore, the probability of a group of n agents to collaborate

is identical to the probability of a individual agent to collaborate. In all other cases (5) allows us to calculate the micro-macro matching from $\mathscr{S}(n)$ to $P(n)$.

A caveat of (5) is that the Normal approximation yields poor results for small n, usually smaller than 20, and is better when $\mathscr{S}(x)$ is neither close to 0 or 1 [6]. In these cases, exact solutions for $P(n)$ require numerical solutions of (4) using what is known as *continuity correction* [12].

4 Microscopic Model

As the proposed collaboration mechanism are strongly non-linear, we chose microscopic stochastic simulations to explore the underlying dynamics of the system. The approach followed to build the stochastic Gillespie simulation of the system is as follows.

- Perform random walk till a collaboration site is found (*search* state).
- Perform algorithm, Task_Allocation (see Algorithm 1) (*wait* state).
- Complete collective task and disperse. (*collaborate* state).
- Return to search.

The probabilistic finite state machine that describes individual agent behavior for this swarm system is shown in Fig. 3. From the individual agent's perspective only one state each exists for *wait* and *collaborate*. From a probabilistic modeling perspective, the wait and collaborate states are meta states, divided into m states each, one for each collaboration site in the arena. This is done to clarify that the probability of collaborating at a given site *only* depends on the number of agents at that specific site and collaborations *only* happen between agents at the same site.

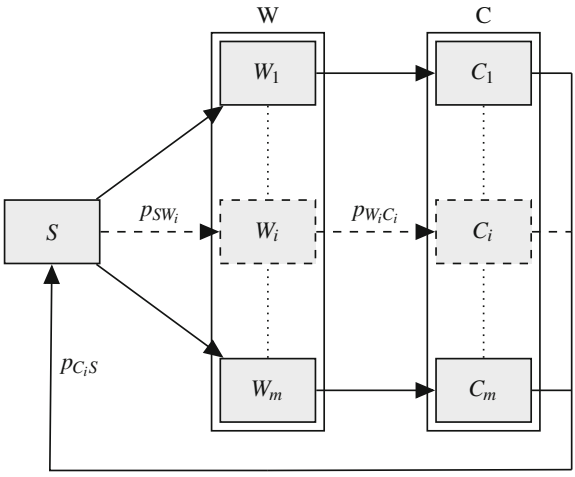


Fig. 3 Agent controller used to drive group collaboration. There is a search state and m wait and collaboration states, W_i and C_i respectively—one for each collaboration site

The probability p_{SW_i} in the PFSM model of the system shown in Fig. 3 is the probability that an agent encounters a collaboration site. This is geometrically computed as the ratio between the total area of the search space (arena) and the total area of collaboration sites, i.e. $p_{SW_i} = n_s(A_s)/A$ (n_s = number of sites, A_s = area per site). The probability, $P_{W_iC_i}$, of going from a wait state to a collaboration state is given by Eq. (4) with input N_{W_i}, the number of agents at collaboration site-i. P_{C_iS} stochastically models the time it takes for an agent to complete a generic collaborative task and is equal to $1/T$, where T is the amount of time (on average) that it takes an agent to complete the collective task. Note that agents have a zero probability of transitioning from the *wait* state back to the *search* without collaborating, i.e. once an agent is at a collaboration site, it will not leave till a collaboration event happens at that site. We chose the following numerical values for all simulations, unless otherwise noted: $A = 100$ and $A_s = 10\,\text{cm}^2$.

For the sake of simplicity, consensus between agents—i.e. going from W_i to C_i—at the same collaboration site is assumed to happen instantly and therefore the extra state(s) is/are omitted from robot controller.

In order to compare the dynamics of the proposed probabilistic task allocation mechanism with the deterministic one by Lerman et. al [18], we implemented a variation of the above algorithm using a unit-step at τ instead of the sigmoid function and removing the consensus step, which is not necessary in this model.

We use Gillespie simulation [14] to explore the dynamics of the proposed collaboration model. For both experiments a single collaboration site is used and each run simulates 300 s of time. The desired group size (τ, in Eq. 1) is set to 4, 8, 16 and 32 agents out of a total of 100 robots. The collaboration task is programmed to take 10 s, on average, per agent. Data points are gathered by averaging data from 100 identically set up runs in each case. The *rate of collaboration* for the threshold model is computed by summing the number of groups that successfully collaborate and dividing by the total experiment time (300 s). For the deterministic model, collaboration rate is computed by summing all successful collaborations, i.e. collaborations involving team sizes equal to τ, and dividing the the experiment time (300 s).

5 Results

We will first compare the dynamics of the proposed approach with Lerman et al.'s k-collaboration model [18] and then validate the emergence of group sizes with similar means but varying variances.

Figure 4a shows collaboration rates for both models when θ is set to 2 (for the probabilistic model) and the wait time is set to ∞ (for the deterministic model), in order to allow for a fair comparison. (All experiments are run in a regime where infinite wait times are optimal wait times, i.e., there are more agents than collaboration sites.) Figure 4a, b and c show collaboration rates for $\theta = 2$, $\theta = 1$ and $\theta = 0$ with infinite wait time. With $\theta = 0$, the Logistic function is uniformly 0.5, allowing any

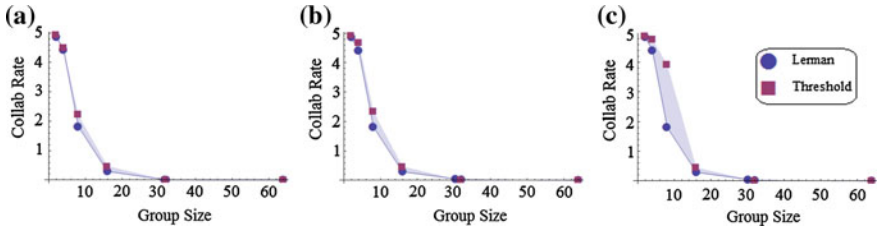


Fig. 4 Comparison of the collaboration rate for task allocation with probabilistic and deterministic [18] for different values of θ and team sizes τ in an environment with one collaboration site and one hundred robots. **a** $\theta = 2$. **b** $\theta = 1$. **c** $\theta = 0$

team size to form. With increasing θ the Logistic function approximates a unit step, minimizing the variance.

We observe the collaboration rate to be qualitatively and quantitatively very similar for high values of θ (steep slope), and to exceed that of the deterministic model for very low values of θ (flat slope). This is expected as flat slopes increase the variance of the observed group size and therefore allow much smaller teams than τ agents to collaborate.

Figure 5 shows histograms of the resulting group sizes for various values of $\tau = 4, 8, 16, 32$ and $\theta = [0; 0.1; 1]$ (100 simulations per data point). It is clearly seen that when θ is set to 0, the sigmoid becomes constant ($\mathscr{S}(x) = 1/(1 + e^0) = 0.5$) so agents have an equal probability to want to collaborate or not, no matter what the desired group size is. We therefore see a large number of small groups forming, with most groups consisting of 2 agents. This is to be expected since the expected number of agents willing to collaborate in a group of size 2 is 1, given the probability of collaboration is constant at 0.5.

Figure 6a displays average group sizes as θ is varied from 0 to 1 and τ from 4 to 32 based on the data from Fig. 5. We observe that for large enough values of θ the mean of the group size distribution approaches the desired group size and is largely unaffected by increasing θ. Thereafter, its magnitude depends only on τ except in the special case where $\theta = 0$ where it is constant. The relative error of the mean compared to the desired average decreases with increasing number of agents as the Binomial distribution (4) approximates the Normal distribution (5).

Figure 6b shows how the variance of group size decreases with increasing θ. This is because the sigmoid function approximates the unit step, making the team size more and more deterministic. On the other hand, low values of θ lead to large variances in the group size. For $\theta = 0$, the variance is constant for all values of τ and depends exclusively on the total number of robots.

Finally, we use the Droplet swarm robot platform to perform real experiments to study the effects of using the proposed task allocation scheme on a physical system. The Droplets are small individually simple robots capable of omni-directional motion and communication (via IR) as well as sensing patterns projected from above. In our experiment we assume that all agents have already arrived at a collaboration site and

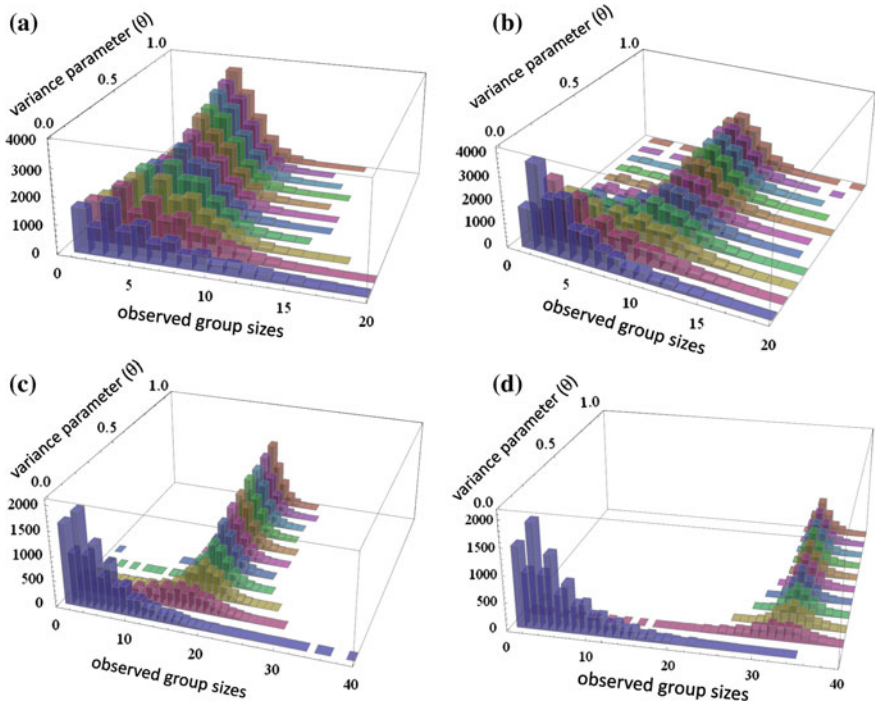


Fig. 5 Histograms of resulting team sizes for various values of τ and θ with one hundred robots and one collaboration site. **a** $\tau = 4$. **b** $\tau = 8$. **c** $\tau = 16$. **d** $\tau = 32$

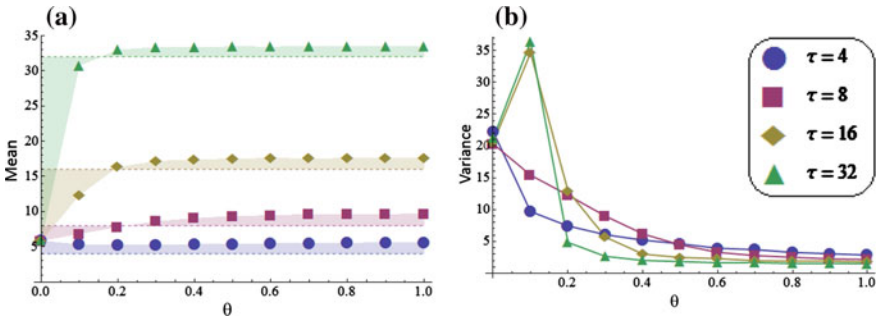


Fig. 6 Showing the effects of varying θ on means and variances corresponding to the histograms seen in Fig. 5

measure the corresponding collaboration rates for a team of 6 robots while varying values of τ and θ. Each agent is individually running the algorithm described in Algorithm 1. A collaboration event is recognized by having all the robots turn on their green LEDs for 5 s. After such a collaboration event, each agent resets its group size estimate and runs Algorithm 1 again.

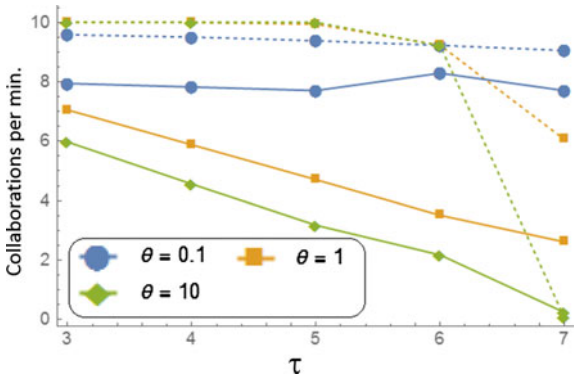


Fig. 7 The *solid lines* show collaborations per min, over 15 min, for a group of 6 real robots as the desired group size is varied from 3 to 7 and the slope of $\mathscr{S}$ is varied between 0.1, 1 and 10. The *dashed lines* indicate simulation results with the same parameters

We ran 5 repeated experiments for all 15 combinations of $\tau = 3, 4, 5, 6$ and 7, and $\theta = 0.1$, 1 and 10, totally 75 runs. Each experiment lasted 15 min and an overhead camera system was set up to detect collaboration events using the software *RoboRealm*. The collaboration rate was a value computed by counting the number of collaborations over the course of each 15 min experiment, normalizing to collaborations per minute, and averaging over the 5 repeated runs. To account for the vision software's detection errors, the raw data gathered from each experiment was de-bounced and passed through a low-pass filter to expose real collaboration events while eliminating observation error. The results of these experiments are seen in Fig. 7. While results are in accordance with simulation for θ being low, the collaboration rate on the real robot platform is much lower than expected for larger θ as simulation assumes perfect communication and group size estimates.

6 Discussion

Results in Figs. 4, 5 and 6 show that the proposed threshold-based task allocation mechanism is a generalization of the deterministic Lerman model in that it allows to approach what is seen with deterministic group sizes while retaining the elasticity to vary group sizes along any desired range of values. Also, these plots show how altering microscopic control parameters within the agents, θ and τ of their sigmoid functions, directly affects macroscopic behavior of the swarm system by altering means and variances of formed group sizes, respectively. Although the matching between microscopic results and macroscopic prediction is not perfect due to the discrete approximation, the plots show that a wide range of means and variances are feasible. Finding appropriate parameters to reach these could be easily achieved using a suitable optimization framework such as presented in [4, 8], using the macroscopic predictions as initial estimate.

The proposed task allocation algorithm requires an estimate of the group size at each collaboration site as well as the ability to communicate with the group in order

to reach a consensus. While these assumptions seem to be limiting at first sight, they can be rolled into the analysis process and possibly exploited to design the task allocation process. For example, an increasing variance for observing the group size τ or noise in the consensus process simply increase the variance of the task allocation process and could therefore be countered—to some extent—by altering the properties of the response threshold function.

This effect is clearly observed in the physical experiment results (see Fig. 7). Since the communication between real robots is not perfect, they almost always underestimate the size of their group resulting in lower collaborations for high θ and τ values. As we observe from comparing the micro simulation results—that are modeled with perfect communication—with real experiment data, we observe a large discrepancy when $\theta = 10$. This happens because although individual agents are set up to be in a group of size 6, their estimates for the group size never cross 4 due to imperfect and blocked communication. Coupled with the fact that the sigmoid threshold effectively acts as a step function when $\theta = 10$, this results in approx. 0 probability of collaboration between agents for a desired group size of 6 but a group size estimate of $\leq$4. Lower values of θ result in better matching between real and simulation data since lower slopes effectively increase the variance in allowed group sizes and mitigate this effect.

We note that there is no optimal wait time as in stick pulling-like collaboration [18]. This optimum exists in swarms with less robots than sticks, which is shown analytically in [19]. Such an optimum does not exist in the proposed model as there is a non-zero probability team sizes with $n < \tau$ will eventually collaborate. Indeed, Algorithm 1 eventually completes as $\mathscr{S}(x) > 0 \, \forall \, x$, i.e., even if only very few robots are at a collaboration site and τ is large, there is a non-zero probability that half or more of the agents at the site eventually collaborate (see also Eq. 4).

Although the algorithm does not deadlock—the probability to collaborate even if the team size is far off the desired value—the resulting behavior might be undesirable, resulting in potentially very long wait times and poor task performance. This could be mitigated by introducing preferential detachment from small groups and preferential attachment to larger groups as customary in swarm robotic aggregation [9].

In practice, effective collaboration rates will also be limited by the embodiment of the robots, which might make finding physical space at a site cumbersome. In the presented microscopic simulation, for both stochastic and deterministic team sizes, the number of robots per site were not limited, allowing scenarios in which multiple groups collaborate in quick succession at the same site. While comparing both models without embodiment is reasonable, we wish to study the effect of embodiment in future work.

7 Conclusion

This paper introduces a task allocation algorithm that allows to recruit an approximate number of agents with a desired variance to a task. This allows to trade-off task execution accuracy with speed, resulting in increasing collaboration rates for teams

with larger variances. We demonstrate that task allocation of teams with deterministic size is a subset of the proposed stochastic task allocation mechanism. As such, the proposed framework provides a computationally simple, adaptive, and robust alternative for coordination.

We investigate the limitations of the proposed approach, which shows lesser fidelity in macroscopic predictions if team sizes are small. In future work, we wish to investigate the impact of variance in estimating the number of agents waiting at a collaboration site as well as the impact of unrel iable communication between agents, both of which we conjecture to impact the variance of resulting team sizes in a similar way as the slope of the response threshold function. We are also interested in studying preferential attachment/detachment techniques from swarm robotic aggregation in order to improve scenarios with insufficient numbers of robots for strong teams to emerge.

Acknowledgments This research has been supported by NSF grant #1150223. We are grateful for this support.

References

1. Amstutz, P., Correll, N., Martinoli, A.: Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm. Ann. Math. Artif. Intell. **52**(2–4), 307–333 (2008)
2. Beni, G.: From swarm intelligence to swarm robotics. In: Swarm Robotics, pp. 1–9. Springer (2005)
3. Beni, G., Wang, J.: Swarm intelligence in cellular robotic systems. In: Robots and Biological Systems: Towards a New Bionics?, pp. 703–712. Springer (1993)
4. Berman, S., Halász, Á., Hsieh, M.A., Kumar, V.: Optimized stochastic policies for task allocation in swarms of robots. IEEE Trans. Robot. **25**(4), 927–937 (2009)
5. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence, From Natural to Artificial Systems. Oxford University Press, New York (1999)
6. Box, G.E.P., Hunter, W.G., Hunter, J.S., et al.: Statistics for Experimenters. Wiley, New York (1978)
7. Chen, J., Sun, D.: Resource constrained multirobot task allocation based on leader-follower coalition methodology. Int. J. Robot. Res. **30**(12), 1423–1434 (2011)
8. Correll, N.: Parameter estimation and optimal control of swarm-robotic systems: a case study in distributed task allocation. In: IEEE International Conference on Robotics and Automation (ICRA) (2008)
9. Correll, N., Martinoli, A.: Modeling and designing self-organized aggregation in a swarm of miniature robots. Int. J. Robot. Res. **30**(5), 615–626 (2011)
10. Dantu, K., Berman, S., Kate, B., Nagpal, R.: A comparison of deterministic and stochastic approaches for allocating spatially dependent tasks in micro-aerial vehicle collectives. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 793–800 (2012)
11. Farrow, N., Klingner, J., Reishus, D., Correll, N.: Miniature six-channel range and bearing system: algorithm, analysis and experimental validation. In: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong (2014)
12. Feller, W.: On the normal approximation to the binomial distribution. Ann. Math. Stat. **16**(4), 319–329 (1945)

13. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
14. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comput. Phys. **22**(403), 403–434 (1976)
15. Klingner, J., Kanakia, A., Farrow, N., Reishus, D., Correll, N.: A stick-slip omnidirectional drive-train for low-cost swarm robotics: mechanism, calibration, and control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2014)
16. Krieger, M.J.B., Billeter, J.-B., Keller, L.: Ant-like task allocation and recruitment in cooperative robots. Nature **406**(6799), 992–995 (2000)
17. Krishnanand, K.N., Amruth, P., Guruprasad, M.H., Bidargaddi, S.V., Ghose, D.: Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In: Proceedings of 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, pp. 958–963. IEEE (2006)
18. Lerman, K., Galstyan, A., Martinoli, A., Ijspeert, A.: A macroscopic analytical model of collaboration in distributed robotic systems. Artif. Life **7**, 93–375 (2001)
19. Martinoli, A., Easton, K., Agassounon, W.: Modeling swarm robotic systems: a case study in collaborative distributed manipulation. Int. J. Robot. Res. **23**, 415–436 (2004)
20. Martinoli, A., Mondada, F., Collective and cooperative group behaviours: biologically inspired experiments in robotics. In: Proceedings of the Fourth International Symposium on Experimental Robotics, pp. 3–10. Springer (1995)
21. Mather, T.W., Hsieh, M.A., Frazzoli, E.: Towards dynamic team formation for robot ensembles. In: IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 4970–4975. IEEE (2010)
22. Sugawara, K., Correll, N., Reishus, D.: Object transportation by granular convection using swarm robots. In: Distributed and Autonomous Robotic Systems (DARS) (2012)
23. Vig, L., Adams, J.A.: Coalition formation: from software agents to robots. J. Intell. Robot. Syst. **50**(1), 85–118 (2007)

Potential Game-Theoretic Analysis of a Market-Based Decentralized Task Allocation Algorithm

Han-Lim Choi, Keum-Seong Kim, Luke B. Johnson and Jonathan P. How

Abstract This paper presents a potential game-theoretic interpretation and analysis of a decentralized task allocation algorithm, consensus-based bundled algorithm, which was developed by the authors' prior work. It is, in particular, proved that the consensus-based bundle algorithm converges to a pure strategy Nash equilibrium of some distributed welfare game, and the price of anarchy and the price of stability of this equilibrium are 1/2 and 1, respectively.

Keywords Task allocation · Potential game · Multi-robot planning · Decentralized planning · Cooperative control

1 Introduction

1.1 Multi-assignment Task Allocation

Consider a multi-robot multi-assignment task allocation that assigns a sequence of tasks to robots to maximize the mission performance metric expressed as sum of rewards from assigned tasks.

H.-L. Choi (✉) · K.-S. Kim
Department of Aerospace Engineering, KAIST, 291 Daehak-ro, Yuseong, Daejeon 305-701, Korea
e-mail: hanlimc@lics.kaist.ac.kr

K.-S. Kim
e-mail: kskim@lics.kaist.ac.kr

L.B. Johnson · J.P. How
Department of Aeronautics and Astronautics, MIT, 77 Massachusetts Ave., Cambridge, MA 02139, USA
e-mail: lbj16@mit.edu

J.P. How
e-mail: jhow@mit.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_15

$$\begin{aligned} \max_{x_{rn}} \ & \sum_{r\in\mathcal{R}}\sum_{n\in\mathcal{N}} s_{rn}(\mathbf{p}_r(\mathbf{x}_r))x_{rn} \\ & \sum_{r\in\mathcal{R}} x_{rn} = 1 \\ & \sum_{n\in\mathcal{N}} x_{rn} \le L \\ & x_{rn} \in \{0,1\},\ \forall (r,n) \in \mathcal{R}\times\mathcal{N} \end{aligned} \tag{P}$$

where $\mathcal{R}$ is the set of robot indices, and $\mathcal{N}$ is the set of task indices. s_{rn} is the reward robot r can obtain by performing task n, along path $\mathbf{p}_r$ consisting of sequence of tasks in $\mathbf{x}_r \triangleq \{x_{rn} : n \in \mathcal{N}\}$. As this score of doing a task is a function of other tasks assigned to the same robot, this s_{rn} can be considered as *some type of marginal* reward from the task n—this paper intentionally does not call this as marginal reward, because such term is reserved for the marginal contribution concept in the context of distributed welfare game. The decision variable $x_{rn} = 1$ if robot r is assigned to n and zero otherwise. The constraints are to ensure that (a) one task can be assigned to only one robot, and (b) one robot can be assigned to a set of (or a sequence of) maximum L tasks.

Assumption 1 As indicated in the formulation in (**P**), this paper assumes that there exists a deterministic scheme that uniquely defines the path ($\mathbf{p}_r$) associated with the assignment ($\mathbf{x}_r$). In other words, given set of tasks to be done, a robot is assumed to be able to uniquely determine the order of the tasks. Note that the notion of path in this context of task allocation indicates the temporal precedence relation rather than the spatial trajectory of the robot. Thus, the uniqueness assumption on the path means that for a given list of tasks to do, a robot uniquely determines the time of execution of those tasks.

Notice that the marginal reward s_{rn} does depend on the assigned tasks to robot r. Giving several examples of reward would be useful to clarify the meaning of this marginal reward. Consider time-critical service tasks come up at some physical locations in the mission space. Then, the reward for the task can be modeled as a time-discounted reward

$$s_{rn} = s_n^0 g_d(t_{rn})$$

with some monotonically decreasing function g_d, where s_n^0 is the base score of task n and t_{rn} is the arrival time of robot r at task n. In this case, the arrival time may depend on the other tasks for the robot to visit. Denote the arrival time for the case where the robot performs only a single task n as $t_{rn}[\{n\}]$. If the robot needs to visit another task k as well as n, then the robot may not be able to arrive at n at $t_{rn}[\{n\}]$. For this particular example of time-discounted reward, if the robot visits k first, $t_{rn}[\{n,k\}] \ge t_{rn}[\{n\}]$; otherwise, $t_{rn}[\{n,k\}] = t_{rn}[\{n,k\}]$.[1] The implication of

[1] This inequality/equality relation may not always hold; for the case of time-windowed, adjustment of the arrival time of the first-visiting task may be needed to make both tasks.

Assumption 1 in this example is that the arrival times are uniquely determined for give set of tasks for a robot to visit.

In some other problems, the definition of the marginal reward may not be very clear as the time-discounted reward. Consider tasking of sensory robot to make observation of some physical phenomena at some regions of interest. If the phenomena at region n is correlated with that at another region k, then resulting entropy reduction typically satisfies:

$$S_r(\{n,k\}) = s_{rn}(\{n\}) + s_{rk}(\{k\}) - \rho_{r,nk}$$

where $S_r(\{n,k\}$ is the total reward from n and k, and $\rho_{r,nk}$ represents the amount of information contained in n to infer about k (or vice versa). In this case, definition of marginal reward $s_{rn}(\{n,k\})$ and $s_{rk}(\{n,k\})$ has degree of freedom; one possible way is to set it as $s_{rn}(\{n,k\}) = s_{rn}(\{n\})$ and $s_{rk}(\{n,k\}) = S_r(\{n,k\}) - s_{rn}(\{n\})$, but the other way is also valid. Although not explicit, it is assumed in this work that there exists a consistent mechanism to define the marginal reward so that the sum of individual rewards corresponds to the total reward from the set of tasks.

1.2 Objectives

The present authors' prior work [2] has presented a decentralized scheme for this multi-assignment task allocation problem, termed consensus-based bundle algorithm (CBBA); in the process, it was proved that the solution by this decentralized algorithm is identical to that of a sequential greedy algorithm. This allows for proving the worst-case performance of the algorithm, which is 1/2 of the optimal solution. With this theoretical finding still holding, this paper takes a game-theoretic perspective on the same problem of multi-assignment task allocation.

This paper heavily relies on recent advancements in the theory of potential games. This game concept was first introduced by [12] in the context of congestion games and rigorously formalized by Monderer and Shapley [11], the Nobel Prize Winner in Economics 2012. The link between the potential game theory and cooperative/coordinated control/decision making has been heavily studied by Marden [1, 6–10]. The present paper takes advantage of recent theoretical advancements by Marden's work, in particular, on the distributed welfare game [9], dynamic ordered protocol [10], and state-based potential games [6].

The main purpose of the present paper is: (a) to report on some interesting relation between the multi-assignment task allocation and the distributed welfare game, and (b) to provide more concrete analysis on the solution properties of CBBA [2]. Although not recognized/described in a game-theoretic way, the key concepts/assumptions introduced in the algorithm can directly be interpreted as some notions in potential games. In particular, this paper proves that: (a) CBBA converges to a pure strategy Nash equilibrium of some distributed welfare game, and (b) the

solution exhibits price of anarchy of 1/2, which is equivalent to 50 % optimality guarantee proved in [2] and price of stability of 1, which is newly proved in this paper.

2 Background

2.1 Distributed Welfare Game

The distributed welfare game considers a class of resource allocation problems where there are a set of N agents and set of resources that are shared by the agents. Each agent i possesses an action set $\mathcal{A}_n \subset \mathcal{R}$ where $\mathcal{R}$ is the set of resources.[2] An action profile is presented by an action tuple $\boldsymbol{a} = (a_1, a_2, \ldots, a_N) \in \mathcal{A}$ where the set of action profiles is denoted by $\mathcal{A} := \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$. Allocation $\boldsymbol{a}$ is often represented as (a_i, a_{-i}) where $a_{-i} := (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_N)$, i.e., the action of all agents except agent i in the allocation. For the *separable* welfare functions, the total welfare for assignment $\boldsymbol{a}$ is expressed as the arithmetic sum of individual rewards per resource:

$$W(\boldsymbol{a}) = \sum_{r \in \mathcal{R}} W_r(\{a\}_r) \tag{1}$$

where $W_r : 2^{\mathcal{N}} \to \mathbb{R}_+$ is the welfare function for resource r and $\{\boldsymbol{a}\}_r := \{i \in \mathcal{N} : r \in a_i\}$ is the set of agents using resource r in the allocation $\boldsymbol{a}$. With this separable welfare function, the welfare generated at a particular resource depends only on which agents are currently using that resource.

The local utility that agent i tries to maximize is represented by

$$U_i(a_i, a_{-i}) = \sum_{r \in a_i} f_r(i, \{a\}_r)$$

where $f_r : \mathcal{N} \times 2^{\mathcal{N}} \to \mathbb{R}$ represents the *protocol* at resource r. A fixed set of protocols $\{f_r\}_{r \in \mathcal{R}}$ results in a well defined game for any collection of action sets $\{A_i\}$'s. This game is termed *distributed welfare game* and defined by the tuple $G = \{\mathcal{N}, \mathcal{R}, \{\mathcal{A}_i\}, \{W_r\}, \{f_r\}\}$. This distributed welfare game is known to be a *potential game*, in which local utility is aligned with the global potential function so that self-interested agents lead to global cooperative behavior.

Definition 1 The welfare function is called *submodular* if the following holds:

$$W_r(T_1 \cup \{i\}) - W_r(T_1) \geq W_r(T_2 \cup \{i\}) - W_r(T_2) \tag{2}$$

[2] The action set can be defined over $2^{\mathcal{R}}$ as long as the separability condition in (1) is satisfied.

for all r, for any player sets $T_1 \subseteq T_2 \subseteq \mathcal{N}$. In other words, the welfare for player i would obtain from resource r is diminishing as the player shares the resource with inclusively more other players.

2.1.1 Dynamic Ordered Protocol

Let denote $\mathcal{P}(\mathcal{N})$ represent the set of possible orders (or permutations) of the agent set $\mathcal{N}$. Therefore, an order $p \in \mathcal{P}(\mathcal{N})$ is a vector of length N where each entry of p is associated with a unique entry in $\{1, \ldots, N\}$. Let p^j represent the index of agent j in the order p. For any agent set $\mathcal{S} \subseteq \mathcal{N}$ and agent $i \in \mathcal{S}$ define $N_i(p, \mathcal{S}) := \{k \in \mathcal{S} : p^k \leq p^j\}$ as the set of agents with an index less than the index of agent i. With a specified order, the resource distribution protocol for resource $r \in \mathcal{R}$ can be expressed as

$$f_r^{\text{ORD}}(i, \mathbf{x}_r; p_r) := W_r(\mathcal{N}_i(p_r, \mathbf{x}_r)) - W_r(\mathcal{N}_i(p_r, \mathcal{S} \setminus \{i\})).$$

An ordered protocol assigns each agent a distributed share in accordance with their marginal contribution to the welfare in their respective order.

The *dynamic* ordered protocol allows for some notion of state transition for the order of the form:

$$p_r(t+1) = g_r(p_r(t), \{a(t)\}_r),$$

with some (probabilistic) mapping g. This notion of ordering can be regarded as some "state" variable in the game and the utility function is defined dependent on the state value. An equilibrium, which is some extended version of pure strategy Nash equilibrium, for the game with a dynamic protocol is defined as:

Definition 2 The equilibria of the dynamic order adjustment process is the set of all action state pairs $[\boldsymbol{a}^\star, p^\star]$ such that the following conditions hold:

1. $p_r^\star = g_r(p_r^\star, \{a^\star\}_r)$ for every resource $r \in \mathcal{R}$.
2. $U_i(a_i^\star, a_{-i}^\star; p^\star) = \max_{a_i \in \mathcal{A}_i} U_i(a_i, a_{-i}^\star; p^\star)$ for every agent $i \in \mathcal{N}$.

In other words, at an equilibrium, the state, i.e., the ordering, is invariant, and the action is the pure strategy Nash equilibrium under the given converged equilibrium.

It was shown in [10] that for the dynamic protocol satisfying the following conditions converges to an equilibrium.

C1 Associate with each agent $i \in Z = \{a(t-1)\}_r \cap \{a(t)\}_r$ a unique order $p_r^i(t) \in \{1, \ldots, |Z|\}$ according to the following condition: For any two agent $i, j \in Z$

$$p_r^i(t-1) < p_r^j(t-1) \Rightarrow p_r^i(t) < p_r^j(t).$$

In other words, if two agents are sharing the resource consecutively, the ordering does not change.

C2 Associate with each agent $i \in \{a(t)\}_r \setminus \{a(t-1)\}_r$ a unique ordering $p_r^j(t) \in \{|Z|+1, \ldots, |\{a(t)\}_r|\}$ according to any deterministic rule. In other words, if new agent takes part in the share of the resource, it should take a lower order than the agents already sharing the resource.

It was also noted that with this dynamics,

$$a(t) = a(t-1) \Rightarrow p(t+1) = p(t).$$

In other words, if the assignment converges then the ordering becomes invariant.

Theorem 1 (Theorem 4.1 in [10]) *Let $\mathcal{G}$ be the set of distributed welfare games with submodular welfare functions and dynamic ordered protocols. The equilibria of the dynamic process is nonempty for any game $G \in \mathcal{G}$. Furthermore, the price of anarchy is 1/2 and the price of stability is 1 across the set of games $\mathcal{G}$.*

2.2 Consensus-Based Bundle Algorithm (CBBA)

The consensus-based bundle algorithm (CBBA) [2] is a decentralized task allocation algorithm that provides provably good task assignments for multi-agent, multi-task allocation problems. The algorithmic structure of CBBA is an iterative, two-phase algorithm. These two phases are: a *bundle building* phase where each vehicle greedily generates an ordered bundle of tasks, and a *task consensus* phase where conflicting assignments are identified and resolved through local communication between neighboring agents. These two phases are repeated until the algorithm has reached convergence, which is proven to happen in finite upperbounded time. While the algorithmic details of CBBA can be found in [2], this section summarizes a new description of the algorithm presented in the authors' recent work [5] utilizing the notion of bid space, which allows for clearer explanation of the main results of this paper. Some key notions to be defined are:

- A **bid** is represented as a triple: $S_{rn} = \langle r, n, s_{rn} \rangle$, where i represents the bidding robot's index, j represents the task's index, and s_{rn} represents the bid score for this robot-task pair.
- A **bundle** is an ordered data structure internal to each robot r, $\mathbf{b}_r = \{r_{rn_1}, \ldots, s_{rn_k}\}$ that consists of all k of its current bids. When new bids are made, they are appended to the end of the bundle, thus the order in the bundle reflects the relative age of each bid and thus the *dependency structure* of the bids.
- The **global bid space** is an unordered set of bids, defined as $\mathcal{A} = \{s_{r_1 n_1}, \ldots, s_{r_m n_m}\}$, which contains a globally consistent set of the current winning bids over the team. In the decentralized decision framework, this information is not available to the robots. In actual implementation of CBBA, this entity is not explicitly computed, but this concept is useful to analyze the properties of the algorithm. The procedure of constructing a global bid space (for generalization of CBBA) is given in the authors' recent work [3, 4]

- A **local bid space** $\mathcal{A}_r$ is defined as a set that contains robot r's current local understanding of the global bid space, i.e., local view on the winning robot-task pairs and the corresponding score value. In a fully connected network, $\mathcal{A}_r = \mathcal{A}$ after each consensus phase, but in general, the geometry of agents in the network may lead to information propagation latencies and thus non-identical local bid spaces.

The bidding score s_{rn} is determined by the marginal reward for robot r would obtain from the task n, if the task is added to the robots current bundle (i.e., the list of tasks that have been committed by robot r):

$$s_{rn}[\mathbf{b}_r] = \mathcal{S}_r[\mathbf{b}_r \oplus_{\text{end}}] - \mathcal{S}_r[\mathbf{b}_r] \tag{3}$$

where $\mathcal{S}_r[\mathbf{b}_r]$ is the total reward robot r would obtain by performing all the tasks in $\mathbf{b}_r$, and the binary operator $A \oplus_{\text{end}} B$ denotes appending the entity B at the end of the list A.[3] In the context of robot task planning, the task order in a bundle is not necessarily identical to the (temporal) order of task execution. The order represents the dependency structure in the calculation of marginal scores rather than temporal/spatial precedence.

Assumption 2 (*Diminishing Marginal Gain*) One key assumption in CBBA is that:

$$s_{rn}[\mathbf{b}_r] \geq s_{rn}[\mathbf{b}_r \oplus_{\text{end}} \mathbf{b}] \tag{4}$$

for any ordered list of tasks $\mathbf{b}$. This condition was referred to as *diminishing marginal gain* (DMG) condition in [2], and is the key assumption for proof of convergence and the performance guarantee of CBBA.

The bundle building phase of CBBA, which is run independently for each robot, is summarized in Algorithm 1. In this phase, each robot sequentially add bids to the bundle, by first choosing a candidate task in a greedy manner, and then checking if it can outperform other robots on the task based on its current local bid space $\mathcal{A}_r$.

After the bundle building phase completes, each robot r synchronously shares its current local bid space $\mathcal{A}_i$ with each of its adjacent neighbors. This local bid space, in combination with time-stamp information, is then passed through some decision table (see [2], Table 1 for details) that provides all of the conflict resolution logic to merge local bid spaces. In general, the consensus logic prefers larger and more recent bids. If the consensus phase has occurred more than twice the network diameter times without any bids changing, the algorithm has converged and terminates; if not, each agent re-enters the bundle building phase and the algorithm continues.

[3]Notice the notational difference in the meaning of $s_{rn}(A)$ and $s_{rn}[B]$. The formal indicates the marginal reward of n with the total assignment of A including n, while the second indicates the marginal reward of n when committed on B, which excludes n.

Algorithm 1 CBBA: Bundle Building Phase

(for robot r)

procedure BUILD BUNDLE($\mathcal{A}_r$)
  for all k such that $s_{rn_k} \in \mathcal{A}_r$ **do**
    $\mathcal{A}_r = \mathcal{A}_r \setminus s_{rn_k}$
  end for
  set $\mathbf{b}_r = \emptyset$
  while $|\mathbf{b}_r| < L$ **do**
    for all $n \in \{1, \ldots, N\} \setminus \mathbf{b}_i$ **do**
      $s_{rn} = s_{rn}[\mathbf{b}_r]$,
      $h_{rn} = \mathbb{I}(s_{rn} > s_{r'n}),\ \forall s_{r'n} \in \mathcal{A}_r$
    end for
    $n^\star = \text{argmax}_n\, s_{rn} \cdot h_{rn}$
    $S_{rn^\star} = \langle i, j^\star, s_{rn^\star} \rangle$
    if $s_{rn^\star} > 0$ **then**
      $\mathbf{b}_r = \mathbf{b}_r \oplus S_{rn^\star}$
    else
      break
    end if
  end while
end procedure

2.2.1 Properties of CBBA

Theorem 2 (Theorem 1 in [2]) *With the DMG satisfying scoring functions, the CBBA process over a static communication network with diameter D satisfies:*

- *CBBA produces the same solution as the sequential greedy algorithm with all the local bid spaces are the same over the network.*
- *The convergence time is bounded above by* $\min\{N, L|\mathcal{R}|\} \cdot D$.

Theorem 3 (Theorem 2 in [2]) *Assuming the agents have accurate knowledge of the situational awareness, CBBA guarantees 50 % optimality for the multi-assignment problem with diminishing marginal gain scoring schemes.*

Remark 1 Although the notion of "diminishing marginal gain" is defined with the concept of bundle, which is a *ordered* list of tasks a certain robot is assigned to, with some repeatable mechanism of bundle construction rules, the same notion can be defined and described for the unordered set functions. Then, this DMG notion is equivalent to the concept of submodularity that has been investigated in many contexts including the distributed welfare game submodularity.

3 Multi-assignment Task Allocation as Distributed Welfare Game

3.1 Resource Allocation Formulation

It should be noted that the formulation in (**P**) that assigns a set of tasks to a robot can be viewed as allocation of sharable resources, i.e., robots, to multiple agents that want to be serviced, i.e., tasks. In this sharing process, agent n obtains some portion of total resource, if it takes resource r. Let define $a_{nr} \in \{0, 1\}$ be one if agent n takes resource r and zero otherwise, and w_{nr} be the portion of resource r allotted to agent n. Also, each resource can be allotted to at most L agents, and one agent cannot utilize multiple resources. Then, the resource allocation problem to maximize the total resource utilization can be written as:

$$\begin{aligned} \max_{a_{nr}} & \sum_{r\in\mathcal{R}} \sum_{n\in\mathcal{N}} w_{nr}(\mathbf{a}_r) a_{nr} \\ & \sum_{r\in\mathcal{R}} a_{nr} = 1 \\ & \sum_{k\in\mathcal{N}} a_{nr} \le L \\ & a_{nr} \in \{0, 1\}\ \forall (n, r) \in \mathcal{N} \times \mathcal{R} \end{aligned} \tag{DP}$$

where $\mathbf{a}_r = \{a_{nr},\ n \in \mathcal{N}\}$. Note that with

$$a_{nr} = x_{rn}, \qquad w_{nr} = s_{rn},$$

the resource allocation formulation in (**DP**) is identical to the original multi-assignment task allocation in (**P**).

3.2 Interpretation as Distributed Welfare Game

For the resource allocation in (**DP**), the welfare from each resource r can be defined xsas:

$$W_r(\boldsymbol{a}_r) = \sum_{n\in\boldsymbol{a}_r} w_{nr} = \sum_{n\in\mathcal{N}} w_{nr} a_{nr}, \tag{5}$$

where $\boldsymbol{a}_r = \{n \in \mathcal{N} : a_{nr} = 1\}$. Then, the total welfare can be expressed the sum of the welfare from individual resources:

$$W(\boldsymbol{a}) = \sum_{r\in\mathcal{R}} W_r(\boldsymbol{a}_r) \tag{6}$$

where $\boldsymbol{a} = (a_1, \ldots, a_N)$ with $a_n \in \mathcal{R}$ being the action of agent n.

Notice that for the multi-assignment task allocation problem considered in this paper, each agent can utilize only one resource and thus the domain of agent action is the set of resources (as opposed to the power set of $\mathcal{R}$. The local utility of agent n can be defined as:

$$U_i(a_n, a_{-n}) = f_r(n, \boldsymbol{a}_r), \tag{7}$$

with some protocol f_r at resource r. Different types of protocols can defined for the welfare function (5), but the natural choice from the original definition of the task allocation problem is to conform the protocol to the marginal reward defined for the allocation problem:

$$f_r(n, \boldsymbol{a}_r) = w_{nr}(\boldsymbol{a}_r) = s_{rn}(\mathbf{x}_r). \tag{8}$$

It should be noted that the protocol in (8) is by construction *budget-balanced* in the sense that:

$$\sum_{n \in \boldsymbol{a}_r} f_r(n, \boldsymbol{a}_r) = W_r(\boldsymbol{a}_r). \tag{9}$$

Therefore, the resource allocation problem in (**DP**) can be interpreted as a distributed welfare game with the welfare functions in (6), the local utility in (7), and the budget-balanced distribution protocol in (8).

As described in the original formulation in (**P**), the score value s_{nr} is defined as the marginal reward the robot would get by performing task n. The notional of "marginal" inherits some notion of dependency structure. As discussed in Sect. 1.2, this dependency structure is closely related to the notion of ordered protocol in the distributed welfare game. In other words, the marginal reward can be expressed as:

$$s_{rn}(\boldsymbol{a}_r) = W_r(\mathcal{N}_i(p_r, \boldsymbol{a}_r)) - W_r(\mathcal{N}_i(p_r, \boldsymbol{a}_r \setminus \{n\})). \tag{10}$$

4 CBBA as Dynamic Ordered Protocol

First notice that the bid calculation in (3) in the bundle building phase of CBBA can be regarded as the computation of the marginal reward with some ordering protocol as the bids are computed with a recursive structure.

For example, for bundle $\mathbf{b}_r$, we can straightforwardly define the ordering such that

$$p_r^{b_{ri}} = i, \ \forall i \leq |\mathbf{b}_r|. \tag{11}$$

In other words, the task (or agent) located at i-th position in robot r's bundle takes the order i. This (simple-looking) ordering is valid, because the CBBA bundle building phase appends a new task at the end of the current bundle, conforming the dependency structure between the task scores. The order of the tasks that are not in $\mathbf{b}_r$ can be defined some distinctive indexing between $|\mathbf{b}_r| + 1$ and N.

The ordering is not invariant throughout the CBBA process, because via the plan consensus phase, robots may change/update their local bid spaces. If a robot drops a task from its bundle by realizing that there exists a better robot that can service the task, then it drops all the bids added later than this outbid bid; the original ordering structure is no longer valid and start adding new task at the dropped task. Although the ordering protocol is varying over the iteration, in the end CBBA converges and no changes are made on each robot's bid space (and on the bundle).

4.1 Sequential Greedy Solution Equivalence

One way to prove the equivalence of the CBBA process to the distributed welfare game with the dynamic ordered protocol is to first take advantage of the convergence proof of the CBBA to the centralized sequential greedy solution. In other words, by showing that the centralized greedy selection process can be interpreted as a legitimate dynamic ordered protocol, we can indirectly prove the equivalence of the CBBA to the DWG. More direct way that constructs the global bid space directly in the procedure of the CBBA is given in the next section.

To recollect the equivalence of the CBBA solution and the sequential greedy solution, [2] proved that with the score functions with diminishing marginal gain, equivalently submodular reward function, the CBBA converges to a solution that is identical to the sequential greedy solution obtained by Algorithm 2.

Algorithm 2 Sequential greedy algorithm [2]

$\mathcal{R}_1 = \mathcal{R},\ \mathcal{N}_1 = \mathcal{N}$
$\eta_i = 0,\ \forall i \in \mathcal{I}$
$s_{rn}^{(1)} = s_{rn}[\{\emptyset\}],\ \forall (r,n) \in \mathcal{R} \times \mathcal{N}$
for $m = 1$ to $\min\{|\mathcal{N}|, L|\mathcal{R}|$ **do**
  $(r_m^\star, n_m^\star) = \text{argmax}_{(r,n)\in\mathcal{R}\times\mathcal{N}}\, s_{rn}^{(m)}$
  $\eta_{r_m^\star} = \eta_{r_m^\star} + 1$
  $\mathcal{N}_{m+1} = \mathcal{N}_m \setminus \{n_m^\star\}$
  $\mathbf{b}_{r_m^\star}^{(m)} = \mathbf{b}_{r_m^\star}^{(m-1)} \oplus_{\text{end}} \{n_m^\star\}$
  $\mathbf{b}_r^{(m)} = \mathbf{b}_r^{(m-1)},\ \forall r \neq r_m^\star$
  if $\eta_{r_m^\star} = L$ **then**
    $\mathcal{R}_{m+1} = \mathcal{R}_m \setminus \{r_m^\star\}$
    $s_{r_m^\star,n}^{(m+1)} = 0, \forall n \in \mathcal{N}$
  else
    $\mathcal{R}_{m+1} = \mathcal{R}_m$
  end if
  $s_{r,n_m^\star}^{(m+1)} = 0,\ \forall r \in \mathcal{R}_{m+1}$
  $s_{rn}^{(m+1)} = c_{ij}[\mathbf{b}_r^{(m)}],\ \forall (r,n) \in \mathcal{R}_{m+1} \times \mathcal{N}_{m+1}$
end for

Lemma 1 (Adapted from Lemma 2 in [2]) *The CBBA process with synchronized consensus phase over a static network of diameter D, every agent agrees on the first k SGA assignments by iteration k D. In other words, defining the set of bids consisting of these SGA assignments:*

$$\mathcal{A}_k^{SG} = \{\langle r_m^\star, n_m^\star, s_{r_m^\star, n_m^\star}^{(m)} \rangle, \quad m \leq k\},$$

then, the global bid space contains this SGA bids:

$$\mathcal{A}_k^{SG} \subseteq \mathcal{A}(kD),$$

so does every robot's local bid space:

$$\mathcal{A}_k^{SG} \subseteq \mathcal{A}_r(kD), \ \forall r \in \mathcal{R}.$$

Consider a dynamic ordered protocol aligned with the sequential greedy recursion. In other words, the state transition of the dynamic ordered protocol occurs every iteration of the SGA procedure. At iteration m, robot r's bundle is $\mathbf{b}_r^{(m)}$; consider its bundle at the next SGA iteration (equivalently next stage of the distributed welfare game). Depending on the winning robot of the $(m+1)$-th SGA robot-task pair, there could be two cases:

$$\mathbf{b}_r^{(m+1)} = \begin{cases} \mathbf{b}_r^{(m)} & \text{if } r_{m+1}^\star \neq r \\ \mathbf{b}_r^{(m)} \oplus_{\text{end}} \langle r, n_{m+1}^\star, s_{r,n_{m+1}^\star} \rangle & \text{if } r_{m+1}^\star = r \end{cases} \tag{12}$$

as the SGA bundle does not delete the task already included. Since the score update rules in lines 12, 16, and 17 allow for adding a task that gives the largest marginal score (conditioned on the previously selected tasks), the following protocol can naturally be defined for this SGA procedure.

$$p_r^{b_{ri}}(m) = \begin{cases} i & \text{if } i \leq |\mathbf{b}_r^{(m)}| \\ |\mathbf{b}_r| + \texttt{idx}(1, |\mathcal{N}| - |\mathbf{b}_r|) & \text{if } i > |\mathbf{b}_r^{(m)}| \end{cases} \tag{13}$$

where $\texttt{idx}(m_1, m_2) \in \{m_1, \ldots, m_2\}$ denotes a consistent assignment of indices between m_1 and m_2. Notice that for the tasks allocated to robot r at both m and $(m+1)$-th iteration, the ordering is invariant; thus, the relative ordering requirement in **C1** in Sect. 2.1.1 is satisfied. Also, for newly added tasks the above protocol provides a valid deterministic ordering. Thus, the second requirement in **C2** is also satisfied. Therefore, the SGA selection procedure is equivalent to a distributed welfare game with the dynamic protocol in (13).

Claim 1 There exists a distributed welfare game with valid dynamic ordered protocol that is equivalent to the sequential greedy algorithm.

4.2 Properties

Thus, at converged CBBA solution, there is one agreed upon ordering structure agreed upon over the network. This allows for proving that the CBBA solution is a pure strategy Nash equilibrium of a state-based potential game with a certain ordered protocol.

Lemma 2 *The CBBA solution with submodular reward function is a pure strategy Nash equilibrium of the distributed welfare game defined in Sect. 3.2 with the following converged ordered protocol:*

$$p_r^{b^\star_{ri}} = \begin{cases} i & \text{if } i \leq |\mathbf{b}_r^\star| \\ |\mathbf{b}_r| + \texttt{idx}(1, |\mathcal{N}| - |\mathbf{b}_r^\star|) & \text{if } i > |\mathbf{b}_r^\star| \end{cases} \tag{14}$$

where $\mathbf{b}_r^\star$ *denotes the bundle of robot r for the converged CBBA solution.*

Proof It suffices to show that assignment of a task k to another robot always decreases marginal reward of the task under the ordering (14). Suppose that task n is assigned to robot r and the (r, n) robot-task pair is the kth sequential greedy solution. By construction, it is true that

$$s_{rn}[\mathbf{b}_r^{(k-1)}] \geq s_{r'n}[\mathbf{b}_{r'}^{(k-1)}], \qquad r' \in \mathbf{R},$$

because otherwise r' must have allocated to task n. From the submodularity

$$s_{r'n}[\mathbf{b}_{r'}^{(k-1)}] \geq s_{r'k}[\mathbf{b}_{r'}^{(k-1)} \cup \{\mathbf{b}\}], \qquad r' \in \mathbf{R}.$$

for any bundle (i.e., list of bids) $\mathbf{b}$. If the ordering upto k is fixed, the incumbent ordering provides the largest marginal reward for the task n. By the definition of unilateral variation, tasks other than n are assigned to the original robots, which means that tasks from the 1st through $k-1$th solution are still assigned to the same robots. Under the same ordering, assignment of n other agent always decreases the total reward.

Theorem 4 *The CBBA solution guarantees the price of anarchy of 1/2 and the price of stability of 1. In other words,*

$$\min \frac{\mathbf{CBBA}}{\mathbf{OPT}} = \frac{1}{2}, \qquad \max \frac{\mathbf{CBBA}}{\mathbf{OPT}} = 1,$$

where **CBBA** *and* **OPT** *denote the total welfare by the CBBA solution and the optimal solution, respectively. This not only means that the optimality of CBBA is bounded below and above but also means that those bounds are attainable.*

Proof The proof is straightforward by noting that the CBBA solution is a pure strategy Nash equilibrium (Lemma 2) and the CBBA process can be interpreted as a valid dynamic ordered protocol (Claim 1).

5 Conclusions

A game-theoretic reformulation of the multi-assignment task allocation problem was presented, in which the game players are the set of tasks (rather than the robots) and the solution properties of a distributed greedy assignment algorithm, specifically the consensus-based bundle algorithm, were analyzed. The analysis has proved that the converged solution from the algorithm is a Nash equilibrium with price of anarchy 1/2 and price of stability 1. Future work will investigate an algorithm advancement on the decentralized greedy selection process to develop a procedure to incrementally converges to the optimal solution.

Acknowledgments This work was supported in part by the Agency for Defense Development (Contract # UE123026JD) and in part by the KI project through KAIST Institute of Design of Complex Systems.

References

1. Arslan, G., Marden, J.R., Shamma, J.S.: Autonomous vehicle-target assignment: a game theoretical formulation. ASME J. Dyn. Syst. Measur. Control **129**(5), 584–596 (2007)
2. Choi, H.L., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Trans. Robot. **25**(4), 912–926 (2009). doi:10.1109/TRO.2009.2022423
3. Johnson, L., Choi, H.L., How, J.P.: Using non-submodular score functions in decentralized task allocation. IEEE Trans. Robot. (submitted)
4. Johnson, L.B., Choi, H.L., How., J.P.: Convergence analysis of the hybrid information and plan consensus algorithm. In: American Control Conference (ACC) (2014)
5. Johnson, L.B., Choi, H.L., Ponda, S.S., How., J.P.: Allowing non-submodular score functions in distributed task allocation. In: IEEE Conference on Decision and Control (CDC) (2012)
6. Marden, J.R.: State based potential games. Automatica **48**(12), 3075–3088 (2012)
7. Marden, J.R., Arslan, G., Shamma, J.: Cooperative control and potential games. IEEE Trans. Syst. Man Cybern. Part B Cybern. **39**(6), 1393–1407 (2009)
8. Marden, J.R., Arslan, G., Shamma, J.: Joint strategy fictitious play with inertia for potential games. IEEE Trans. Autom. Control **54**(2), 208–220 (2013)
9. Marden, J.R., Wierman, A.: Distributed welfare games. Oper. Res. **61**(1), 155–168 (2013)
10. Marden, J.R., Wierman, A.: Overcoming the limitations of utility design for multiagent systems. IEEE Trans. Autom. Control **58**(6), 1402–1415 (2013)
11. Monderer, D., Shapley, L.S.: Potential games. Games Econ. Behav. **14**, 124–143 (1996)
12. Rosenthal, R.: The network equilibrium problem in integers. Networks **3**(1), 53–59 (1973)

The Hybrid Information and Plan Consensus Algorithm with Imperfect Situational Awareness

Luke Johnson, Han-Lim Choi and Jonathan P. How

Abstract This paper presents an extension to the Hybrid Information and Plan Consensus Algorithm (HIPC) that accounts for imperfect situational awareness (SA). This algorithm uses implicit coordination to plan for a subset of the team on-board each agent, then uses plan consensus to satisfy assignment constraints. By combining the ideas of implicit coordination and local plan consensus, the algorithm empirically reduces the convergence time for distributed task allocation problems. The contribution of this work is that it extends previous results to account for the likely possibility of imperfect situational awareness across the team. This is accomplished by tracking when predictions are incorrect and removing offending predictions if they are hindering algorithmic convergence. Empirical results are provided to demonstrate that this new approach allows the use of inconsistent situational awareness to improve convergence speed.

Keywords Task allocation · Distributed planning · Multi-agent systems

1 Introduction

The goal of standard multi-agent task allocation algorithms [1, 2] is to coordinate a team of cooperative agents to achieve an overall mission objective. These mission objectives can often be broken up into tasks that require agents to visit certain locations or regions, all while minimizing some resource usage (i.e. fuel, power, etc.) Centralized solutions are typically preferred when an application requires high levels of collaboration. However, in contested environments where communications

L. Johnson (✉) · J.P. How
Department of Aeronautics and Astronautics, MIT, Cambridge, MA, USA
e-mail: lbj16@mit.edu

J.P. How
e-mail: jhow@mit.edu

H.-L. Choi
Division of Aerospace Engineering, KAIST, Yuseon, Daejeon, Korea
e-mail: hanlimc@kaist.ac.kr

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_16

may be unavailable, unreliable, have high latency, or high cost, relying on centralized solutions can be impractical. In these communication limited environments, it is necessary to consider distributed or decentralized algorithms [3]. Unfortunately, using distributed or decentralized algorithms usually introduces additional complications, including difficulties establishing both algorithmic convergence and performance. The major reason for these complications is that in decentralized environments, agents may operate on only partial information, and thus independent agent optimizations may not align perfectly with each another. To correct for this misalignment, advanced communication protocols are typically required for decentralized algorithms. These communication protocols can utilize two different information assumptions: global information consistency and local information consistency.

1. *Global information consistency assumptions* require that **all** agents agree upon certain relevant pieces of information during the task allocation algorithm's execution. This agreement forms a set of "correct" information that agents can independently recognize as team-wide truth. Given that these global information consistency assumptions require recognizing information across the entire team, reaching information consistency happens on a global communication time scale. Algorithms that utilize these assumptions can be found in Refs. [4–23].
2. *Local information consistency assumptions* do not require global consistency of any information, however, it is still assumed that information will propagate to the entire team eventually (just not by any particular time.) Only requiring local information consistency can provide a much shorter time-scale for utilizing new information (than global information consistency), because agents are not required to ensure that this information has propagated to the entire team before using it. The natural downside of this is that agents cannot guarantee any piece of information is globally consistent and thus algorithms utilizing these assumptions must take this into account during the planning process [24–27].

Additionally, the type of information communicated in task allocation algorithms can traditionally be divided into two different paradigms: (1) implicit coordination [5, 7, 28], and (2) plan consensus [17, 18, 24–27, 29, 30]. Implicit coordination task allocation algorithms require that all agents in a multi-agent team come to consensus on all parameters that would be relevant to solving the planning problem (this set of information will be called situational awareness). The consensus process then provides each agent the independent capability to compute the full team-wide assignment. Implicit coordination requires global information consistency assumptions to guarantee a consistent view of world for all agents. In fact, when agents do not have a consistent view of the world, (i.e. under local information consistency assumptions) it is possible to experience an arbitrarily large performance gap.

In plan consensus [17, 18, 22–27, 29, 30], all agents are allowed different views of the world, but communication is explicitly used to enforce consistency of the allocation assignments. This paradigm works well when agents are specialized (in location or capability,) such that assignment conflicts are minimized. Plan consensus algorithms can be specialized to operate either with global information consistency

assumptions such as in [17, 18, 22, 23] or with local information consistency assumptions [24–27].

The insight used in this paper is that the two paradigms can have complementary advantages if care is taken to properly integrate them. With implicit coordination, agents may plan in what are effectively locally centralized teams to provide highly coupled allocations. Conversely, plan consensus can be used to enforce assignment constraints that may have been violated due to mismatches in the situational awareness across the team. Combining algorithms in each paradigm is a complex endeavor because the integration requires that each paradigm know exactly what the others provide. A naive combination of these paradigms in the best case may lead to an inefficient use of computation and communication resources and, in the worst case, can lead to arbitrarily bad task allocations. Previous work [31, 32] developed an algorithm called the hybrid information and plan consensus (HIPC) algorithm which used local information consistency assumptions to provide provable performance and convergence guarantees if individual agents obtained perfect situational awareness over subsets of the team. This work extends the previous work to incorporate imperfect situational awareness in the planning problem, providing a non-trivial extension of previous work that involves adding algorithmic mechanisms to *detect* when using incorrect information is actually reducing planner performance.

2 Problem Statement

This section presents the general problem statement and formalizes the class of problems being solved by HIPC. Given a set of N_a agents and N_t tasks, the goal of the task allocation algorithm is to find an assignment of tasks to agents that maximizes the global reward. The global objective function for the mission is given by a sum over local objective functions for each agent, while each local reward is determined as a function of the tasks assigned to that agent, and the times at which those tasks will be serviced. This task assignment problem can be written as the following mixed-integer (possibly nonlinear) program:

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a} \sum_{j=1}^{N_t} F_{ij}(\mathbf{x}, \boldsymbol{\tau})\, x_{ij} \\ \text{s.t.} \quad & \mathbf{H}(\mathbf{x}, \boldsymbol{\tau}) \leq \mathbf{d} \\ & \mathbf{x} \in \{0,1\}^{N_a \times N_t}, \ \boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t} \end{aligned} \tag{1}$$

where $\mathbf{x} \in \{0,1\}^{N_a \times N_t}$, is a set of $N_a \times N_t$ binary decision variables, x_{ij}, which are used to indicate whether or not task j is assigned to agent i; $\boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a \times N_t}$ is the set of real-positive decision variables τ_{ij} indicating when agent i will service its assigned task j (where $\tau_{ij} = \emptyset$ if task j is not assigned to agent i); F_{ij} is the score function for agent i servicing task j given the overall assignment; and $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 \dots \mathbf{h}_{N_c} \end{bmatrix}^T$, with $\mathbf{d} = \begin{bmatrix} d_1 \dots d_{N_c} \end{bmatrix}^T$, define a set of N_c possibly nonlinear

constraints of the form $\mathbf{h}_k(\mathbf{x}, \boldsymbol{\tau}) \leq d_k$ that capture transition dynamics, resource limitations, etc. This problem formulation can accommodate most design objectives and constraints used in multi-agent decision making problems (e.g. search and surveillance missions where F_{ij} represents the value of acquired information and the constraints $\mathbf{h}_k$ capture fuel limitations and/or no-fly zones, or rescue operations where F_{ij} is time-critical favoring earlier τ_{ij} execution times, etc.). An important observation is that, in Eq. (1), the scoring and constraint functions explicitly depend on the decision variables $\mathbf{x}$ and $\boldsymbol{\tau}$, which makes this general mixed-integer programming problem (NP-hard) [33].

The algorithms used in this paper solve *distributed greedy multi-agent multi-assignment* problems. For each agent, these problems take a similar form to Eq. (1), except individual agents independently create assignments for themselves, and then iterate with others using a consensus algorithm to produce a final team-wide assignment. The details of this process will be explored throughout the rest of this paper.

3 HIPC Overview

The Hybrid Information and Plan Consensus (HIPC) algorithm is a distributed algorithm that provides task assignments for multi-agent, multi-task allocation problems. The algorithmic structure of HIPC is an iterative, 2 phase algorithm. These two phases are: a *local bid space creation* phase where each agent generates a personal allocation of tasks (possibly using situational awareness of other agents in the team), and a *task consensus* phase where conflicting assignments are identified and resolved through local communication between adjacent agents. These two phases are repeated until the algorithm converges. To further explain the relevant details of the algorithm, some notation will be formalized.

3.1 HIPC Notation

Bid A bid is represented as a triple: $s_{ij} = \langle i, j, c \rangle$, where i is the bidding agent's index, j is the bid's task id, and c represents the bid score.

Bundle A bundle is an ordered data structure internal to each agent i, $\mathbf{b}_i = \{s_{ij_1}, \ldots, s_{ij_n}\}$ that consists of all n of its current bids. When new bids are made, they are appended to the end of the bundle, thus the order in the bundle reflects the relative age of each bid and thus the *dependency structure* of the bids, where later bids scored depend on the assignment of all earlier bids in the bundle.

Bid Space The bid space is an unordered set of bids, defined as

$$\mathcal{A} = \{s_{i_1 j_1}, \ldots, s_{i_N j_N}\},$$

where N is the index of the last element in the bid space. This set contains a globally consistent set of the current winning bids in the fleet.

Algorithm 1 HIPC for each agent i

```
1: procedure HIPC(A_i^0, J, N_i)
2:     set A_i = A_i^0, A'_i = A_i^0
3:     iterationNumber = 1
4:     while convergenceCounter < 2D do
5:         (A_i, N_i) = CreateHIPCAssignment(A'_i, N_i)
6:         A'_i = Consensus Phase(A'_i, A_i)
7:         if A'_i = A_i then
8:             convergenceCounter = convergenceCounter + 1
9:         else
10:            convergenceCounter = 0
11:        end if
12:        iterationNumber = iterationNumber + 1
13:    end while
14: end procedure
```

Local Bid Space A local bid space $\mathcal{A}_i$ is defined as a set that contains agent i's current local understanding of the global bid space. In a fully connected network, $\mathcal{A}_i = \mathcal{A}$ after each communication phase, but in general, the geometry of agents in the network may lead to information propagation latencies and thus non-identical local bid spaces.

Network Diameter The network diameter, $\mathcal{D}$, is defined as the communication distance between the furthest agents in the communication network. More formally, define the communication distance between any pair of agents i and i' to be $\mathcal{D}_{i\to i'}$. Define $\mathcal{D} = \max_{i,i'} \mathcal{D}_{i\to i'}$ to be the maximum communication distance over all agent pairs i and i'.

Neighborhood The neighborhood, $\mathcal{N}_i$, of an agent i is defined as the set of agents that agent i has situational awareness over. Similarly an agent i's exact neighborhood $\bar{\mathcal{N}}_i$ is the set of agents that agent i has *perfect* situational awareness over. By definition, an agent's perfect neighborhood always includes itself ($i \in \bar{\mathcal{N}}_i$) and an agent's perfect neighborhood is always a subset of its full neighborhood ($\bar{\mathcal{N}}_i \subseteq \mathcal{N}_i$). If an agent $i' \in \mathcal{N}_i$, then agent i expects that it can predict the objective values for agent i'. This does not mean that their task allocations will be trivially identical even when perfect information is known because neither their neighborhoods ($\mathcal{N}_i \neq \mathcal{N}_{i'}$) nor their local bid spaces ($\mathcal{A}_i \neq \mathcal{A}_{i'}$) will be identical in general.

3.2 HIPC Algorithmic Description

The high level HIPC algorithmic description is given in Algorithm 1.

1. HIPC is a procedure run on-board each agent independently. HIPC is initialized with an initial bid space $\mathcal{A}_i^0$, an available task set $\mathcal{J}$, and a set of agents $\mathcal{N}_i$ that

Algorithm 2 Creating the HIPC Assignment i

1: **procedure** CREATEHIPCASSIGNMENT($\mathcal{A}'_i, \mathcal{N}_i$)
2: $\hat{\mathcal{N}}_i \leftarrow \emptyset$
3: **while** $\hat{\mathcal{N}}_i \neq \mathcal{N}_i$ **do**
4: $\hat{\mathcal{N}}_i \leftarrow \mathcal{N}_i$
5: $\hat{\mathcal{A}}_i \leftarrow \mathcal{A}'_i$
6: **for all** $s_{i'j} \in \hat{\mathcal{A}}_i$ s.t. $i' \in \bar{\mathcal{N}}_i$ **do**
7: $\hat{\mathcal{A}}_i \leftarrow \hat{\mathcal{A}}_i \notin s_{i'j}$
8: **end for**
9: $\hat{\mathcal{A}}_i \leftarrow TAA(\hat{\mathcal{A}}_i, \mathcal{N}_i)$
10: $\hat{\mathcal{N}}_i =$ **CheckSAConsistency**$(\hat{\mathcal{A}}_i, \mathcal{A}'_i, \mathcal{N}_i)$
11: **end while**
12: **Return** $(\hat{\mathcal{A}}_i, \hat{\mathcal{N}}_i)$
13: **end procedure**

each agent i has situational awareness over (Line 1). Note that $\mathcal{N}_i$ will contain a set of agents $\bar{\mathcal{N}}_i$ that agent i knowingly has perfect situational awareness over.

2. Before each bid space construction operation, each agent checks for convergence. If $\mathcal{A}_i$ hasn't changed for two times the network diameter ($2\mathcal{D}$) number of iterations, then the algorithm has converged (Line 4).
3. The algorithm calls the `CreateHIPCAssignment` subroutine. The objective of this function is to take in the current local bid space $\mathcal{A}'_i$ and the neighborhood set $\mathcal{N}_i$ and compute an updated local bid space $\mathcal{A}_i$ and potentially an updated neighborhood set $\mathcal{N}_i$ (Line 5). The neighborhood set would only be updated if agent i decided that planning for some agent i' was hindering algorithmic performance and it dropped i' from $\mathcal{N}_i$. The exact details of this subroutine are outlined in Algorithm 2.
4. The next algorithmic step involves running the consensus phase where each agent i, shares its personal bundle $\mathbf{b}_i$ with its network neighbors (Line 6). This consensus phase can be implemented identically to the one proposed in [26]. (If the network is static, other consensus protocols exist where each agent only needs to share the changes to $\mathbf{b}_i$ reducing overall communication. If the communication network is non-static during plan construction, the full $\mathbf{b}_i$ will need to be shared to retain the worst convergence bounds.)
5. The algorithmic convergence condition is checked (Line 7), if it is true then the $convergenceCounter$ is incremented by one (Line 8), if not $convergence - Counter$ is reset to zero (Line 10).
6. Lastly the iteration counter is incremented (Line 12) and the algorithm returns to Line 4.

3.2.1 Creating the HIPC Assignment

The following section describes Algorithm 2 which dictates how the local bid space $\mathcal{A}$ and the neighborhood $\mathcal{N}_i$ are incrementally updated inside the HIPC algorithm.

Algorithm 3 Checking SA consistency with assignments i

1: **procedure** CHECKSACONSISTENCY($\mathcal{A}_i$, $\mathcal{A}'_i$,$\mathcal{N}_i$)
2: **for all** $i' \in \mathcal{N}_i \setminus \bar{\mathcal{N}}_i$ **do**
3: $\bar{s}_{i'j} = \max(s_{i'j})$ s.t. $s_{i'j} \in \mathcal{A}_i$, $s_{i'j} \notin \mathcal{A}'_i$
4: $\gamma \leftarrow \langle \cdot, \bar{s}_{i'j} \rangle$
5: **if** $\gamma \in \Gamma_{i'}$ **then**
6: **if** $z + \mathcal{D}_{i \to i'} + \mathcal{D}_{i' \to i} < iterationNumber$ **then**
7: $\mathcal{N}_i \leftarrow \mathcal{N}_i \setminus i'$
8: continue
9: **end if**
10: **else**
11: $\Gamma_{i'} = \Gamma_{i'} \cup \langle iterationNumber, \bar{s}_{i'j} \rangle$
12: **end if**
13: **end for**
14: **Return** $(\mathcal{N}_i)$
15: **end procedure**

1. The procedure takes in the output from consensus at the previous iteration $\mathcal{A}'_i$ and the current neighborhood set $\mathcal{N}_i$ (Line 1).
2. The first time the algorithm reaches $\hat{\mathcal{N}}_i \neq \mathcal{N}_i$, (Line 3), the while loop will trivially return true ($|\mathcal{N}_i| \geq 1$). Each subsequent time through (Line 3) checks if the neighborhood set has changed, when it hasn't, the procedure returns.
3. Drop all tasks from the local bid space $\hat{\mathcal{A}}_i$ that belong to agents in known perfect situational awareness set $\bar{\mathcal{N}}_i$ (Lines 6–8).
4. Use the TAA to compute an updated local bid space, $\hat{\mathcal{A}}_i$. New bids are computed locally for every agent in the neighborhood set $\mathcal{N}_i$ (Line 9). The default behavior inside of the TAA function is that agents cannot remove any bids that they have already committed to, they can only bid on unassigned tasks, or outbid a current winner of a task. Note that all bids made by agents in $\bar{\mathcal{N}}_i$ are removed before entering TAA, so the bundles for these agents are built up from scratch.
5. Update neighborhood set $\hat{\mathcal{N}}_i$ from the output of `CheckSAConsistency` (Line 10).
6. Return new local bid space $\hat{\mathcal{A}}_i$ and new neighborhood set $\hat{\mathcal{N}}_i$ (Line 12).

3.2.2 Checking Situational Awareness Consistency

The following section describes Algorithm 3 which describes how agents i' are removed from agent i's neighborhood $\mathcal{N}_i$.

1. The procedure takes in the new predicted bid space $\mathcal{A}_i$ the output from consensus at the previous iteration $\mathcal{A}'_i$ and the current neighborhood set $\mathcal{N}_i$ (Line 1).
2. Iterate over all agents that i has imperfect SA over (Line 2).
3. Find the max bid predicted for agent i' that is in the new bid space $\mathcal{A}_i$ but not in the old one $\mathcal{A}'_i$ (Line 3)

4. Check if this bid is in agent i's previous overbid list for agent i', defined as $\Gamma_{i'}$ (Line 5). $\Gamma_{i'}$ consists of pairs $\langle z_{\bar{s}_{i'j}}, \bar{s}_{i'j} \rangle$ where $z_{\bar{s}_{i'j}}$ is the iteration number where $\bar{s}_{i'j}$ was added to $\Gamma_{i'}$.
5. Check if the pair $\langle z_{\bar{s}_{i'j}}, \bar{s}_{i'j} \rangle$ has been in the previous overbid list for longer than a communication loop to the agent being planned for ($\mathcal{D}_{i \to i'} + \mathcal{D}_{i' \to i}$) (Line 6), and if the predicted bid has been in local bid space too long remove i' from $\mathcal{N}_i$ (Line 7) and continue checking other agents (Line 8).
6. If overbid is new, construct a pair of *iterationNumber* and bid $\bar{s}_{i'j}$ and add to $\Gamma_{i'}$ (Line 11).
7. Return updated neighborhood $\mathcal{N}_i$ (Line 14).

3.2.3 Task Assignment Algorithm (TAA)

This implementation of HIPC can utilize any centralized task allocation algorithms with a few required features (Line 9 from Algorithm 2).

- Bids on separate tasks must follow a bundle structure. Specifically, this means that bids are created in an ordered way, where the value of each bid is based on acyclic dependencies. In CBBA, bundles are constructed as ordered lists, where bids at later places in the list are dependent on the assignment of every previous task in the list. This can be implemented with a generalized score function by value of the most recent bid must be the incremental value of adding it to the overall collection of personal tasks.
- The value of tasks shared with other agents is monotonically decreasing with respect to their dependency structure. i.e. if a bid $\mathbf{s}_1$ is dependent on a bid $\mathbf{s}_2$, then $c_1 \leq c_2$. This can also accommodate arbitrary score functions by using a process called Bid Warping described in [29].
- Agents cannot remove any bids that they have already committed too inside of the TAA, they can only bid on unassigned tasks, or outbid a current winner of a task.

The implementation of HIPC in this paper uses a centralized implementation of CBBA [26], but this is not required and may not be desired for some objective functions.

3.3 Convergence and Performance Insights

Due to space limitations the full convergence and performance proofs are not provided in this paper. Instead the key ideas of the proofs will be provided below with their main result. The full convergence result is only a slight modification of the result published in [32].

Theorem 1 *HIPC converges in at most $2N_t(N_a + 1)\mathcal{D}$ iterations, where N_t is the number of available tasks, and N_a is the number of agents in the team.*

Proof Roughly, each agent can only mispredict the next task to be *locked in*, (see [32] for a precise definition of this) for $2\mathcal{D}$ iterations. This misprediction can cycle through every agent for a maximum of $2N_a\mathcal{D}$ number of iterations. At this point, the next task to be *locked in* will be bid on by its winning agent and this will take $2\mathcal{D}$ iterations to propagate for a total of $2(N_a + 1)\mathcal{D}$ iterations to lock in each task. However, every agent now has reduced the cardinality of their neighborhood set $\mathcal{N}$ by 1. The full allocation will then be realized in at most $2N_t(N_a + 1)\mathcal{D}$ iterations.

This bound will not be tight unless $N_t = 1$ and would require an incredibly malicious set of circumstances to realize because every single prediction that each agent makes will need to give the worst possible answer (in a convergence sense). If the SA is really this bad, each neighborhood will end up containing only the agents' self and all subsequent replanning iterations will be purely plan consensus. Additionally if $N_t \gg N_a$ (which is normally true) the convergence bound becomes $\approx 2N_t\mathcal{D}$, the perfect SA worst case bound.

Theorem 2 *HIPC with imperfect situational awareness will converge to the same solution as a centralized TAA*

Proof This result follows closely with the convergence proof. The winning bids that a centralized TAA algorithm would return are sequentially *locked in* (again see [32] for a precise definition of this) in order of largest score using exactly the same logic as the convergence proof.

4 Experimental Results

To validate the claimed results in this paper two Monte Carlo experiments were run to demonstrate expected performance. The environments were created in a room resembling the physical flight volume at the Aerospace Controls Lab as can be seen in Fig. 1. In the Monte Carlo experiments run for this paper, both agents and tasks were randomly placed in the room according to a uniform distribution over the open space.

The first Monte Carlo experiment was run where situational awareness was perfectly known for a subset of the fleet. The scenario run in Fig. 2 was with 5 agents and a varying number of tasks for 100 Monte Carlo iterations where the number of iterations are averaged over all of the runs. The parameter "HIPC size" refers to the number of other agents that each agent has perfect situational awareness over (i.e. HIPC size: $(|\tilde{\mathcal{N}}_i| - 1), \forall i$). In this figure a dramatic reduction of convergence iterations can be seen as the HIPC size increases. The blue line in this figure, corresponding to HIPC size: 0 is the result that pure plan consensus would give (exactly the CBBA solution [26]). The magenta line in the figure, corresponding to HIPC size: 4 would be the global implicit coordination solution. All the space between these two lines consist of hybrid solutions. In the experiment the HIPC size was kept

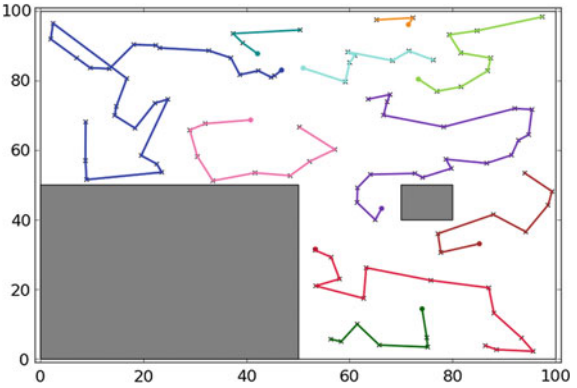


Fig. 1 Figure shows an example of what the planning environment and solution would look like for a random example with 10 agents and 100 tasks. The *small colored circles* in the figure represent the agents initial locations, the x's are the task locations and the *colored line segments* represent the winning agent's estimated trajectory to service its winning bids

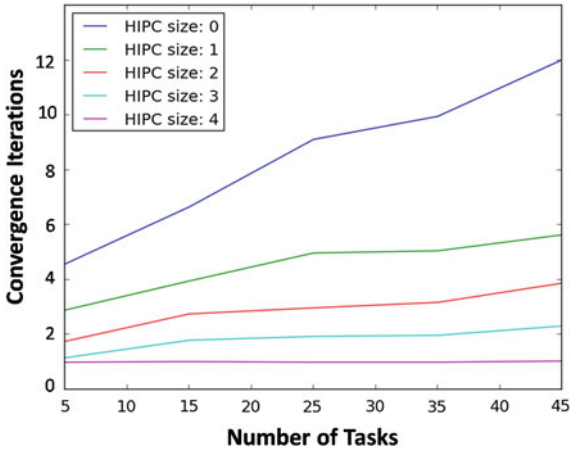


Fig. 2 Figure shows the reduction in convergence iterations introduced by when perfect situational awareness is provided to HIPC. This example is a 5 agent scenario run on a varying number of tasks. HIPC size refers to the number of other agents each agent knowingly has perfect SA over (corresponding to $(|\bar{\mathcal{N}}_i| - 1)$

consistent for all agents, but recall this does not need to be true in practice. The HIPC size can even be asymmetric between agents planning for each other.

The second Monte Carlo experiment highlights the contribution of this paper specifically. The results of this experiment are shown in Fig. 3. This experiment was run on 7 agents with 45 tasks for 300 Monte Carlo iterations where the number of iterations is averaged over all runs. Again in this graph HIPC size refers to the number of other agents each agent is planning for $(|\bar{\mathcal{N}}_i| - 1)$ except in this example situational awareness is imperfect. The x-axis in this figure is called "Starting Location Error". The way that this environment was constructed is at each true location for a given task, a box was centered at this true task location. The imperfect locations were constructed by sampling a 2 dimensional uniform distribution where the edge lengths of the sample regions were "Starting Location Error" times the maximum dimensions of the arena. If multiple agents i were predicting the same agent i', each agent i had a unique sample for the expected location of agent i'. As expected, when the

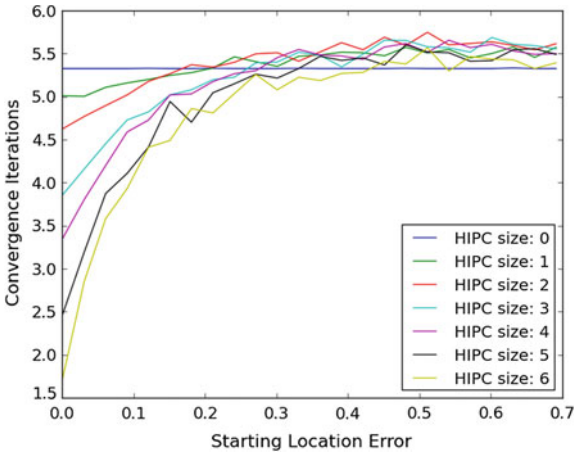


Fig. 3 Figure shows the how the convergence rate of HIPC decays when situational awareness is degraded. The example used in this figure was 7 agents competing over 45 tasks. The starting location error refers to a percentage of the entire arena over which the initial location of agents in $\mathcal{N}_i$ can be wrong. HIPC size refers to the number of other agents, each agent has partial situational awareness over (corresponding to $|\mathcal{N}_i| - 1$)

information is more consistent, planning for more agents reduces the number of convergence iterations dramatically.

An interesting case to consider is when the HIPC size $1 - 6$ lines cross above the (HIPC size: 0) line. The crossing point for each respective line corresponds to when the error in information is bad enough such that not considering any extra information would have led to faster convergence. This increase in average convergence time is a result of cases where the incorrect SA is leading to bad predictions and the agents needed to *learn* to stop planning for those agents. This learning takes more time than the savings due to predictive capability of other parts of the network. In general, missions may have a combination of perfect and imperfect situational awareness and the algorithm would handle this naturally. The convergence speed would be between the extreme cases. Also worth noting here is that for identical problem statements, all "HIPC sizes" and "Starting Location Errors" returned the same allocation with the only difference being the time to convergence. The fact that the solutions converged to the same score was part of the design of the algorithm and is a positive feature of this approach. Worth note is that for both experiments a global consistency assumption planner [22, 23] could be used, and the number of iterations for these planners would be the number of tasks assigned. This would lead to slower algorithmic convergence in each of the two experiments considered in this paper.

5 Conclusion

This paper presents an extension to the Hybrid Information and Plan Consensus Algorithm previously introduced in [31, 32] that accounts for imperfect situational awareness. This algorithm uses implicit coordination to plan for a subset of the team on-board each agent, then uses plan consensus to fix conflicts in the assignment

constraints that may arise. By combining these ideas of local plan consensus and implicit coordination the algorithm empirically reduces the convergence time for distributed task allocation. The algorithm extends previous results to imperfect situational awareness by tracking when predictions are incorrect and removing offending predictions if they are hindering algorithmic convergence. Empirical results demonstrate that this approach allows the use of fairly rough situational awareness to still improve convergence speed.

Acknowledgments This work was sponsored (in part) by the AFOSR and USAF under grant (FA9550-11-1-0134). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

References

1. Ponda, S.S., Johnson, L.B., Geramifard, A., How, J.P.: Cooperative mission planning for Multi-UAV teams. In: Handbook of Unmanned Aerial Vehicles. Springer (2012)
2. Gerkey, B.P., Mataric, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Athena Scientific, MA (1997)
4. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proc. IEEE **94**(7), 1257–1270 (2006)
5. McLain, T.M., Beard, R.W.: Coordination variables, coordination functions, and cooperative timing missions. AIAA J. Guidance Control Dyn. **28**(1), 150–161 (2005)
6. Castanon, D.A., Wu, C.: Distributed algorithms for dynamic reassignment. IEEE Conf. Decis. Control (CDC) **1**(9–12), 13–18 (2003)
7. Curtis, J., Murphey, R.: Simultaneous area search and task assignment for a team of cooperative agents. In: AIAA Guidance, Navigation, and Control Conference (GNC) (2003). (AIAA-2003-5584)
8. Ren, W., Beard, R.W., Kingston, D.B.: Multi-agent Kalman consensus with relative uncertainty. Am. Control Conf. (ACC) **3**(8–10), 1865–1870 (2005)
9. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Autom. Control **49**(9), 1520–1533 (2004)
10. Alighanbari, M., How, J.P.: An unbiased kalman consensus algorithm. In: American Control Conference (ACC), pp. 3519–3524. Minneapolis, 14–16 June (2006). http://acl.mit.edu/papers/ACC06_AlighanbariHow.pdf
11. Moallemi, C.C., Roy, B.V.: Consensus propagation. IEEE Trans. Inf. Theory **52**(11), 4753–4766 (2006)
12. Olshevsky, A., Tsitsiklis, J.N.: Convergence speed in distributed consensus and averaging. IEEE Conf. Decis. Control (CDC) 3387–3392 (2006)
13. Ren, W., Beard, R.W., Atkins, E.M.: Information consensus in multivehicle cooperative control. IEEE Control Syst. Mag. **27**(2), 71–82 (2007)
14. Hatano, Y., Mesbahi, M.: Agreement over random networks. In: 43rd IEEE Conference on Decision and Control (2004)
15. Wu, C.W.: Synchronization and convergence of linear dynamics in random directed networks. IEEE Trans. Autom. Control **51**(7), 1207–1210 (2006)
16. Tahbaz-Salehi, A., Jadbabaie, A.: On consensus over random networks. In: 44th Annual Allerton Conference (2006)

17. Sariel, S., Balch, T.: Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In: AIAA Workshop on Integrating Planning Into Scheduling (2005)
18. Ahmed, A., Patel, A., Brown, T., Ham, M., Jang, M., Agha, G.: Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In: International Conference on Integration of Knowledge Intensive Multi-Agent Systems (2005)
19. Atkinson, M.L.: Results analysis of using free market auctions to distribute control of UAVs. In: AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit (2004)
20. Lemaire, T., Alami, R., Lacroix, S.: A distributed task allocation scheme in multi-UAV context. IEEE Int. Conf. Robot. Autom. (ICRA) **4**, 3622–3627 (2004)
21. Walsh, W., Wellman, M.: A market protocol for decentralized task allocation. Proc. Int. Conf. Multi Agent Syst. (1998)
22. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Proc. Robot. Sci. Syst. (2005)
23. Amstutz, P., Correll, N., Martinoli, A.: Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm. Ann. Math. Artif. Intell. **52**(2–4), 307–333 (2008). http://dx.doi.org/10.1007/s10472-009-9127-8
24. Hoeing, M., Dasgupta, P., Petrov, P., O'Hara, S.: Auction-based multi-robot task allocation in comstar. In: Proceedings of the 6th International Joint Conference on Autonomous agents and multiagent systems (2007)
25. Sujit, P.B., Beard, R., Distributed sequential auctions for multiple UAV task allocation. Proc. Am. Control Conf. (2007)
26. Choi, H.-L., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Trans. Robot. **25**(4), 912–926 (2009). http://dx.doi.org/10.1109/TRO.2009.2022423
27. Marden, J., Wierman, A.: Overcoming the limitations of utility design for multiagent systems. IEEE Trans. Autom. Control **58**(6), 1402–1415 (2013)
28. Shima, T., Rasmussen, S.J., Chandler, P.: UAV team decision and control using efficient collaborative estimation. Am. Control Conf. (ACC) **6**(8–10), 4107–4112 (2005)
29. Johnson, L.B., Choi, H.L., Ponda, S.S., How, J.P.: Allowing non-submodular score functions in distributed task allocation. IEEE Conf. Decis. Control (CDC) (2012). http://acl.mit.edu/papers/cdc_12_cbba_submitted.pdf
30. Johnson, L.B., Ponda, S.S., Choi, H.L., How, J.P.: Improving the efficiency of a decentralized tasking algorithm for UAV teams with asynchronous communications. In: AIAA Guidance, Navigation, and Control Conference (GNC) (2010). http://acl.mit.edu/papers/ACBBA_GNC10_submitted_07272010.pdf
31. Johnson, L., Choi, H.-L., How, J.P.: Hybrid information and plan consensus in distributed task allocation. In: AIAA Guidance, Navigation, and Control Conference (GNC) (2013)
32. Johnson, L., Choi, H.-L., How, J.P.: Convergence analysis of the hybrid information and plan consensus algorithm. In: American Control Conference (ACC), Portland (2014)
33. Bertsimas, D., Weismantel, R.: Optimization Over Integers. Dynamic Ideas Belmont, MA (2005)

Part III
Formation Control and Path Planning

Adaptive Leader-Follower Formation in Cluttered Environment Using Dynamic Target Reconfiguration

José Vilca, Lounis Adouane and Youcef Mezouar

Abstract This paper presents a control architecture for safe and smooth navigation of a group of Unmanned Ground Vehicles (UGV) while keeping a specific formation. The formation control is based on *Leader-follower* and *Behavioral* approaches. The proposed control architecture is designed to allow the use of a single control law for different multi-vehicle contexts (navigation in formation, transition between different formation shapes, obstacle avoidance, etc.). The obstacle avoidance strategy is based on the limit-cycle approach while taking into account the dimension of the formation. A new Strategy for Formation Reconfiguration (SFR) of the group of UGVs based on suitable smooth switching of the set-points (according, for instance, to the encountered obstacles or the new task to achieve) is proposed. The inter-vehicles collisions are avoided during the SFR using a penalty function acting on the vehicle velocities. Different simulations on cluttered environments show the performance and the efficiency of the proposal, to obtain fully reactive and distributed control strategy for the navigation in formation of a group of UGVs.

Keywords Cooperative robots · Autonomous navigation · Formation control · Dynamic formation · Target-reaching control

1 Introduction

In the last decades, research interest in control and coordination of multiple robots has increased significantly. Different tasks that may be performed by a single complex robot can be performed with more flexibility and efficiency by a group of elementary

J. Vilca (✉) · L. Adouane · Y. Mezouar
Institut Pascal, Blaise Pascal University—UMR CNRS, 6602 Clermont-Ferrand, France
e-mail: jose.vilca@univ-bpclermont.fr

L. Adouane
e-mail: lounis.adouane@univ-bpclermont.fr

Y. Mezouar
e-mail: youcef.mezouar@univ-bpclermont.fr

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_17

cooperative robots. These cooperatives robots join their capacities and information to improve the task achievement.

Exploration [16], management and platooning of autonomous vehicles [5], mapping of unknown locations [19], coverage of unknown area [9], transport of heavy objects [3], rendez-vous of multiple agents [21] are some examples of multi-robot tasks (Fig. 1).

Nonetheless, the coordination of multi-robot navigation is among the most challenging tasks, due notably, to its implication for instance for public transportation [15]. This paper addresses the navigation of a group of vehicles in formation (i.e., when a group of mobile robots has to navigate and to keep a desired relative positions to each other or to a reference). Mostly, three approaches have been investigated to deal with this problem: *behavior-based* [22], *virtual structure* [10] and *Leader-follower* [12] approaches. In this work, the proposed control architecture is based on *Leader-follower* and *behavior-based* approaches for the formation control problem.

In the *Leader-follower* approach, the leader is the reference for the desired configuration of the followers. Different works exploit graph theory to describe the inter-vehicle communications [5, 8, 18, 20]. Different formation cases (leader reassignment, robot adding and control saturation) were presented in [18]. The authors proposed a formation control law based on the combination of Linear Matrix Inequalities and hybrid system. The case of dynamic formation, i.e., the formation shape changes to another (e.g. from square to triangle), and obstacle avoidance was dealt in [5, 8, 20]. In [5], the leader generates a free-collision trajectory in a dynamic environment which is tracked using a formation control law based on neural network, Lyapunov function and dynamic model of the robot. The stability of the dynamic formation and dynamic topology (adjacency matrix) are also demonstrated. In [8], switches between different formation shapes are exploited (from triangle to line) to avoid encountered obstacles in the environment. The formation control law is based on input-output feedback linearization and vision sensors (omnidirectional camera) are embedded in each robot for localization and navigation purpose. A strategy to modify the formation configuration by scaling the distance between the vehicles

Fig. 1 Autonomous navigation in formation of a group of UGVs in an urban environment (Clermont-Ferrand, France)

is proposed in [20]. Obstacle avoidance is dealt with using potential fields. In the already presented works, interesting solutions for formation control problem are proposed. Nevertheless, they are based on predefined trajectories and do not address the issues related to the constraints of the formation shape and to the dimension of the vehicles (large vehicles need large space for navigation and obstacle avoidance). In this paper, the main proposal is to present a fully reactive and distributed navigation in formation of a fleet of UGVs. Additionally, we make a focus on the reconfiguration phase based on a suitable smooth switches between the formation shapes while considering the follower configurations.

In this paper, the dynamics of the followers' set-points are given by the specific dynamic of the leader (*Leader-follower* approach). This strategy allows a good flexibility proprieties for the formation shape [7]. The *behavior-based* approach allows to use different elementary controllers to perform several sub-tasks (cf. Fig. 2). The obstacle avoidance controller for the whole formation is based on limit-cycle principle [1]. This control architecture is designed in order that each follower tracks safely the assigned configuration (given by the leader) while using an appropriate dynamic target-reaching controller. The proposed strategy for formation reconfiguration takes into account the presence of obstacles in the environment as well as the inter-vehicles distance to avoid collision between the UGVs.

The rest of this paper is organized as follows: in the next section, the control architecture for the navigation of UGVs in formation is introduced. The model of the UGV and its controllers are also detailed. In Sect. 3, the navigation in formation based on *Leader-follower* approach is described. Moreover, this section presents also the reconfiguration method for the multi-robot formation. Simulations showing the efficiency of the proposed strategy are detailed in Sect. 4. Finally, conclusion and future works are given in Sect. 5.

2 Control Architecture

The control architecture for the navigation in formation of a group of UGVs is shown in Fig. 2. This control architecture is designed for a group of UGVs modeled as tricycle robots. This architecture aims to manage the interactions among elementary controllers while guaranteeing the stability of the overall control [2]. It allows to obtain safe and smooth navigation of the formation (cf. Sect. 3). The global navigation in formation framework is operated by the *Formation parameters* block that sends to each elementary controller (*Dynamic target reaching* and *Obstacle avoidance*) its desired set-points. Each elementary controller (cf. Fig. 2) provides as output a Control Input **CI** to the *Control law* block through a *Hierarchical action selection* block which selects between $\mathbf{CI}_T$ or $\mathbf{CI}_O$.

In this work, a single control law for the UGV (tricycle robot) is used. It considers the vehicle poses and velocities. This control law allows the UGV to reach a static or dynamic target with a desired orientation and velocity (cf. Sect. 2.2.3). The inputs of the control law (pose errors between the vehicle and its assigned target) are

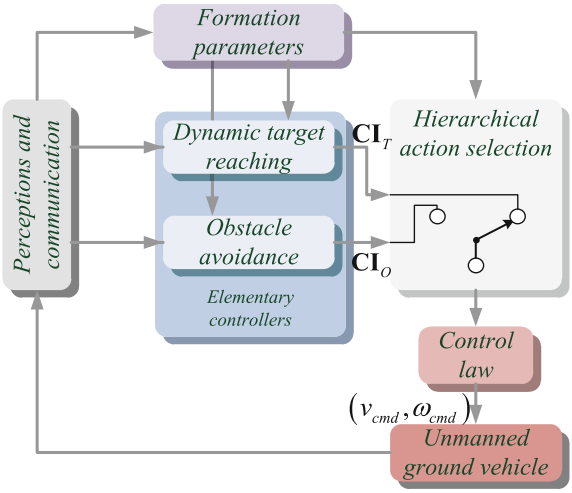


Fig. 2 Proposed control architecture embedded in each UGV navigating in the formation

provided by the elementary controllers (cf. Sect. 2.2). The control law is synthesized according to Lyapunov theorem (more details are given in [23]). The main blocks of the architecture are detailed below.

The *Perceptions and communication* block incorporates the propriocetive and exteroceptive sensors such as range sensor, cameras, odometers and RTK-GPS. Its goal is to capture information related to the robot environment, mainly potential obstacles [6, 8]. The communication is related with the vehicle capability to send and to receive information from other vehicles. In the sequel, we assume a stable communication between UGVs without latency) and that each UGV has a RTK-GPS and a range sensor LIDAR.

The *Formation parameters* block determines the desired configuration for the group of UGVs to keep a specific distance and orientation between them. The Leader-follower approach is used to obtain the formation configuration of the UGVs according to the Leader configuration (cf. Sect. 3). Moreover, a reconfiguration strategy between the formation shapes according to the context of the environment (dynamic and/or cluttered or not) is described in Sect. 3.2.

The control architecture uses a *Hierarchical action selection* mechanism to manage the switches between the two elementary controllers (Behavior-based approach), *Dynamic target reaching* and *Obstacle avoidance* blocks, according to the formation parameters and environment perception. The hierarchical action selection mechanism activates the *Obstacle avoidance* block as soon as it detects at least one obstacle which can hinder the future vehicle movement toward its dynamic virtual target (more details are given in [1]). It allows to anticipate the activation of obstacle avoidance controller and to decrease the time to reach the assigned target (static or dynamic). In order to provide the enough overall details of the presented control architecture, the following subsections present the UGV model and elementary controllers.

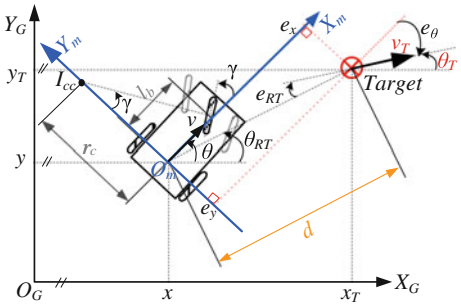


Fig. 3 UGV and dynamic target configuration variables in (local and global) Cartesian reference frames

2.1 Vehicle and Target Set-Point Modeling

We assume that the UGVs evolves in asphalt road and in cluttered urban environment with relatively low speed (less than $v_{max} = 2$ m/s). Hence, the use of kinematic model (which relies on pure rolling without slipping) of the UGV is sufficient. The kinematic model of the UGV is based on the well-known tricycle model [17]. The two front wheels are replaced by a single virtual wheel located at the center between the front wheels. The equations of UGV model can be written as (cf. Fig. 3):

$$\dot{x} = v\cos(\theta); \quad \dot{y} = v\sin(\theta); \quad \dot{\theta} = v\tan(\gamma)/l_b \tag{1}$$

where (x, y, θ) is the UGV pose in the global reference frame $X_G Y_G$. v and γ are respectively the linear velocity and the orientation of the vehicle front wheel. l_b is the wheelbase of the vehicle.

2.2 Elementary Controllers

Each elementary controller generates the control inputs **CI** (pose errors (e_x, e_y, e_θ) and velocities v_T) of the *Control law* block (cf. Fig. 2).

2.2.1 Dynamic Target Reaching Controller

The target set-point modeling is defined by the formation shape (*Formation parameters* block) and it is computed according to the leader configuration (cf. Sect. 3). The target is defined by (x_T, y_T, θ_T) and v_T which are respectively the target poses and linear velocity in the global reference frame. Indeed, an RTK-GPS embedded in each vehicle allows to estimate its current configuration.

Before to introduce the control law, let us describe the following notations (cf. Fig. 3):

- I_{cc} is the instantaneous center of curvature of the vehicle trajectory, $r_c = l_b/\tan(\gamma)$ is the radius of curvature and $c_c = 1/r_c$ is the curvature.
- (e_x, e_y, e_θ) are the errors w.r.t local frame $(X_m Y_m)$ between the vehicle and the target poses.
- θ_{RT} and d are respectively the angle and distance between the target and vehicle positions.
- e_{RT} is the error related to the vehicle position (x, y) w.r.t the target orientation.

This controller guides the vehicle towards the dynamic target. It is based on the pose control of the UGV w.r.t. the target (represented by errors variables (e_x, e_y, e_θ) in Fig. 3). These errors are computed w.r.t the local reference frame $X_m Y_m$ and they are given by:

$$\begin{cases} e_x = \cos(\theta)(x_T - x) + \sin(\theta)(y_T - y) \\ e_y = -\sin(\theta)(x_T - x) + \cos(\theta)(y_T - y) \\ e_\theta = \theta_T - \theta \end{cases} \tag{2}$$

The error function e_{RT} is added to the canonical error system (2) (cf. Fig. 3). Let us now write d and θ_{RT} as (cf. Fig. 3):

$$d = \sqrt{(x_T - x)^2 + (y_T - y)^2} \tag{3}$$

$$\begin{cases} \theta_{RT} = \arctan\left((y_T - y)/(x_T - x)\right) & \text{if } d > \xi \\ \theta_{RT} = \theta_T & \text{if } d \leq \xi \end{cases} \tag{4}$$

where ξ is a small positive value ($\xi \approx 0$). The error e_{RT} is defined as (cf. Fig. 3):

$$e_{RT} = \theta_T - \theta_{RT} \tag{5}$$

Furthermore, the velocity set-points of the dynamic target (in global frame $X_G Y_G$) are computed according to the leader velocities and formation shape (cf. Sect. 3). Finally the pose errors and velocities $(e_x, e_y, e_\theta, v_T)$ are the input of the *Control law* block (cf. Sect. 2.2.3).

2.2.2 Obstacle Avoidance Controller

Different methods can be found in the literature for obstacle avoidance [14, 24]. One of them is the limit-cycle method, the UGV avoids reactively the obstacle if it tracks accurately limit-cycle trajectories as detailed in [1] (ellipse of influence). The main ideas behind this controller are briefly detailed below:

The differential equations of the elliptic limit-cycles are:

$$\dot{x}_s = m(By_s + 0.5Cx_s) + x_s(1 - Ax_s^2 - By_s^2 - Cx_s y_s) \quad (6)$$

$$\dot{y}_s = -m(Ax_s + 0.5Cy_s) + y_s(1 - Ax_s^2 - By_s^2 - Cx_s y_s) \quad (7)$$

with $m = \pm 1$ according to the avoidance direction (clockwise or counter-clockwise). (x_s, y_s) corresponds to the position of the UGV according to the center of the ellipse. The variables A, B and C are given by:

$$A = (\sin(\Omega)/b_{lc})^2 + (\cos(\Omega)/a_{lc})^2 \quad (8)$$

$$B = (\cos(\Omega)/b_{lc})^2 + (\sin(\Omega)/a_{lc})^2 \quad (9)$$

$$C = (1/a_{lc}^2 - 1/b_{lc}^2)\sin(2\Omega) \quad (10)$$

where a_{lc} and b_{lc} characterize respectively the major and minor elliptic semi-axes and Ω gives the ellipse orientation when it is not equal to 0.

Let us now extend this method, initially proposed for the navigation of a mono-robot, to the case of a group of UGVs. In Sect. 3, the formation is defined by longitudinal h_i and lateral l_i coordinates (15) (cf. Fig. 4). At this aim, the dimensions of the ellipse (a_{lc}, b_{lc}) are increased according to the maximum lateral coordinate of the formation shape l_{imax}, i.e., the dimension of the ellipse to avoid will be $(a_{lc} + l_{imax}, b_{lc} + l_{imax})$. The advantage of the proposed method is to maintain the shape of the whole formation even when obstacles hinder the formation navigation, instead of each robot avoids locally the obstacles [2].

In our case, the controller can be written as an orientation control. We consider thus $e_x = 0$ and $e_y = 0$ in (2) (cf. Fig. 3), i.e., the vehicle position is at each sample time in the desired position. The limit-cycle propriety allows to avoid the obstacles. The desired vehicle orientation is given by the differential equation of the limit-cycle (6) and (7):

$$\theta_d = \arctan(\dot{y}_s/\dot{x}_s) \quad (11)$$

Furthermore, the linear velocities of the UGVs are decreased for safe avoidance when the obstacle avoidance controller is activated, e.g., $v_T = v_{max}/2$.

2.2.3 Control Law

The used control law is designed according to Lyapunov stability analysis [23]. The desired vehicle's linear velocity v and its front wheel orientation γ that make the errors (e_x, e_y, e_θ) converge always to zero can be chosen as:

$$v = v_T \cos(e_\theta) + K_x\,(K_d e_x + K_l d \sin(e_{RT})\sin(e_\theta) + K_o \sin(e_\theta)c_c) \quad (12)$$

$$\gamma = \arctan(l_b\left[r_{c_T}^{-1}\cos^{-1}(e_\theta) + c_c\right]) \quad (13)$$

where c_c is given by:

$$c_c = \frac{d^2 K_l \sin(e_{RT}) \cos(e_{RT})}{r_{c_T} K_o \sin(e_\theta) \cos(e_\theta)} + K_\theta \tan(e_\theta) + \frac{K_d e_y - K_l d \sin(e_{RT}) \cos(e_\theta)}{K_o \cos(e_\theta)} + \frac{K_{RT} \sin^2(e_{RT})}{\sin(e_\theta) \cos(e_\theta)} \tag{14}$$

$\mathbf{K} = (K_d, K_l, K_o, K_x, K_{RT}, K_\theta)$ is a vector of positive constants which must be defined by the designer. K_d, K_l and K_o are respectively related to the desired convergence of the distance, lateral and angular errors w.r.t. the assigned target. K_x, K_{RT} and k_θ are related to the maximum linear and angular velocities (more details are given in [23]).

3 Navigation in Formation

We consider a group of N UGVs with the objective of reaching and keeping their assigned configuration according to the desired formation and leader configuration [8, 25]. The proposed strategy consists on controlling each UGV (follower) to track its assigned virtual dynamic target (cf. Sect. 3.1 and Fig. 4). The strategy of formation reconfiguration (cf. Sect. 3.2) is based on a suitable smooth switch of these virtual dynamic targets while considering the inter-vehicle collisions.

3.1 Leader-Follower Approach

Leader-follower approach allows to maintain a rigid geometric shape (e.g. a triangle in Fig. 4). The formation is defined in this case w.r.t. the Cartesian frame (local frame of the leader) (cf. Fig. 4). The proposed formation, based on *Leader-follower* approach, is defined by:

- A leader (UGV$_L$ in Fig. 4); its pose (x_L, y_L, θ_L) and its linear velocity v_L determine the dynamic of the formation (cf. Fig. 4).
- The formation structure is defined with as much nodes as necessary to obtain the desired formation shape. Each node i is a virtual dynamic target (T_{d_i}). The formation is defined as $\mathbf{F} = \{\mathbf{f}_i, i = 1 \ldots N\}$, where $\mathbf{f}_i$ are the coordinates $(h_i, l_i)^T$ of the dynamic target T_{d_i} w.r.t. the leader local reference frame (cf. Fig. 4).

The position and orientation of each node (virtual target) are computed from the leader configuration. The leader position determines the nodes positions according to the formation shape. The instantaneous center of curvature I_{cc_L} of the formation is determined by the leader according to its movements (cf. Fig. 4). I_{cc_L} allows to compute the desired orientation of the nodes according to the formation shape (cf. Fig. 4). The leader turns around I_{cc_L} (positioned perpendicularly to its rear wheel), then the other target set-points T_{d_i} must also turn around I_{cc_L} to maintain the rigid

formation. Thus, the target velocity v_{T_i} must be tangent to the circle which has I_{cc_L} as center and the distance between T_{d_i} and I_{cc_L} as radius $r_{c_{T_i}}$.

The idea behind this strategy is to eliminate the dependency of each UGV to a global reference frame. A straightforward transformation can be applied to obtain the set-point w.r.t. a local reference frame attached to the leader. The polar coordinates (r_i, Φ_i) can also be used by applying a straightforward transformation. An important advantage of the used *Leader-follower* approach is that it does not depend here on any reference trajectory and the formation is fully defined by the instantaneous dynamic of the leader. An important advantage of the proposed formation definition based on *Leader-follower* approach is that it takes, in addition to the target positions (x_T, y_T), the heading θ_T of the virtual targets, which allows to have even more accurate formation navigation (cf. Sect. 4). Furthermore, the proposed approach is more reactive in the sense that it takes at each sample time only the current configuration and velocity of the Leader, instead of using the trajectory of the Leader as a reference for the formation [5, 20].

One important consideration to take into account to achieve the presented navigation of formation strategy, is that the followers have to know, as accurately as possible, the leader state (pose and velocity). As mentioned before, we assume that the leader sends its state by stable Wi-Fi communication without latency. However, cameras and/or LIDAR sensors embedded in each follower, can be used to estimate the leader state [8, 11].

In the sequel, $\mathbf{f}_i$ is given in Global Cartesian frame to homogenize the notation of the equations. The pose of the virtual target T_{d_i} w.r.t the leader pose in the Global reference frame can be written as (cf. Fig. 4):

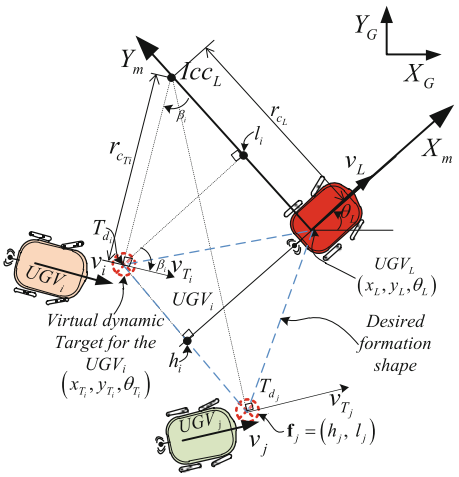


Fig. 4 Formation definition in mobile Cartesian frame linked to the leader

$$\begin{cases} x_{T_i} = x_L + h_i \cos(\theta_L) - l_i \sin(\theta_L) \\ y_{T_i} = y_L + h_i \sin(\theta_L) + l_i \cos(\theta_L) \\ \theta_{T_i} = \theta_L + \beta_i \end{cases} \tag{15}$$

where (x_L, y_L, θ_L) is the current pose of the leader and β_i is the T_{d_i} orientation w.r.t. the leader pose. It is given by:

$$\beta_i = \arctan\left(h_i/(r_{c_L} - l_i)\right) \tag{16}$$

where r_{c_L} is the radius of curvature of the leader. Differentiating (15), the velocities of each T_{d_i} are given thus by:

$$v_{T_i} = \sqrt{(v_L - l_i\omega_L)^2 + (h_i\omega_L)^2} \tag{17}$$

$$\omega_{T_i} = \omega_L + \dot{\beta}_i \tag{18}$$

where v_L and ω_L are respectively the linear and angular velocities of the leader, $\dot{\beta}_i$ is computed as:

$$\dot{\beta}_i = -h_i\dot{r}_{c_L}/\left((r_{c_L} - l_i)^2 + (h_i)^2\right) \tag{19}$$

One can note from (19) that when $\dot{\beta}_i$ is equal to zero, the formation has a constant radius of curvature r_{c_L} and the angular velocities of the virtual targets are equal to the angular velocity of the leader ($\omega_{T_i} = \omega_L$) (18).

3.2 *Proposed Strategy for Formation Reconfiguration*

Different methods dealing with formation reconfiguration for a group of UGV have been proposed in the literature [4, 5, 8]. Many methods exploit Model Predictive Control (MPC) based on time horizon and optimization of a cost function [4]. These methods are generally time consuming due to predictive computation w.r.t. a time horizon. Moreover, they were applied to small unicycle robots and they are based on predefined trajectories computed along a time horizon. This subsection proposes a new Strategy for Formation Reconfiguration (SFR) based on suitable smooth switches between different virtual target configurations (cf. Sect. 3.2.1). This strategy allows to obtain a fully reactive architecture in the sense that the UGV followers track the instantaneous state (pose and velocity) of its virtual targets (thus, without any use of a reference trajectory or a trajectory planning process). Additionally to the reconfiguration process, one should manage potential collisions between UGVs and allocation of virtual targets to UGVs (cf. Sect. 3.2.2).

Different algorithms optimizing target assignment can be easily integrated in our multi-block control architecture (cf. Fig. 2) (refer to [2, 25]). Nonetheless, this paper is focused on the control strategy for formation reconfiguration. Therefore, the allocation of virtual targets to UGVs is achieved using elementary rules when a

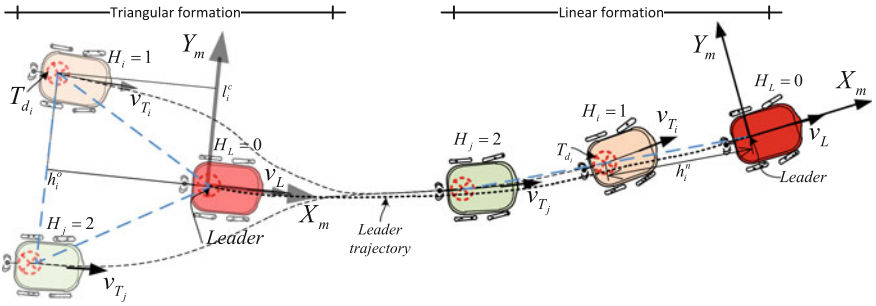


Fig. 5 Formation reconfiguration between, for instance, *triangular* and *linear* formation shapes

formation reconfiguration is required (cf. Sect. 4). These rules assign a label H_i of the virtual target T_{d_i} to the UGV_i at the beginning of the experiments. This label is kept by each UGV along of the reconfiguration process (cf. Fig. 5).

3.2.1 Reconfiguration Method

A typical example of application of formation reconfiguration is when the formation detects a narrow tight corridor, therefore the formation have to adapt to the corridor dimension to continue the navigation. The proposed strategy for formation reconfiguration is based on suitable smooth switching of the virtual target configurations. The new virtual targets defined on the new formation shape must be ahead to the UGVs to guarantee the stability of the overall system (the vehicle must not go back to reach the new virtual target). If this condition is not satisfied then the former formation will be adapted by increasing smoothly and contentiously the longitudinal coordinates h_i until that all UGVs will be positioned in the right configuration (21). The error between the coordinates of the former and the new formation $\mathbf{e}_{f_i}(e_{h_i}, e_{l_i})$ is defined as:

$$\mathbf{e}_{f_i} = \mathbf{f}_i^n - \mathbf{f}_i^f \tag{20}$$

where $\mathbf{f}_i^f(h_i^f, l_i^f)$ and $\mathbf{f}_i^n(h_i^n, l_i^n)$ are respectively the coordinates of the former formation and new desired formation (cf. Figs. 4 and 5).

The reconfiguration process between the different formation shapes is given by:

$$\mathbf{f}_i = \begin{cases} h_i = h_i^n - e_{h_i} e^{-k_r(t-t_r)}, & l_i = l_i^n; \text{ if } e_{h_i} < 0 \\ h_i = h_i^n, & l_i = l_i^n; \text{ if } e_{h_i} \geq 0 \end{cases} \tag{21}$$

where $\mathbf{f}_i(h_i, l_i)$ are the coordinates of the current virtual target T_{d_i} to be tracked by the follower UGV_i. e_{h_i} is the longitudinal coordinate of $\mathbf{e}_{f_i}$ that allows to detect if the virtual target is ahead to its respective follower ($e_{h_i} \geq 0$). The adaptation function when $e_{h_i} < 0$ (virtual target behind to follower$_i$) is set as proportional to the error

between formation shapes, where k_r is a real positive constant designed according to the dynamic of the leader and $t_r > 0$ is the initial time for the reconfiguration process. An accurate analysis of this adaptive reconfiguration function will be developed in future works.

3.2.2 Collision Between UGV

Collision between UGVs can occur during the reconfiguration phase of the group of UGVs. To address this collision risk, we use a penalty function acting on the linear velocity of the UGVs. Moreover, this reduced velocity of UGVs allows to obtain a smooth and less oscillating vehicles' movements (cf. Sect. 4). Each UGV is enclosed by two circle C_{int} and C_{ext} with respectively radius R_{int} and R_{ext} ($R_{int} < R_{ext}$). The collision occurs when the distance d_{ij} between UGV_i and UGV_j are less than R_{int}. Hence, the penalty function ψ_i^j for the UGV_i w.r.t. the UGV_j is defined as:

$$\psi_i^j = \begin{cases} 1 & \text{if } d_{ij} \geq R_{ext} \\ (d_{ij} - R_{int_i})/(R_{ext} - R_{int_i}) & \text{if } R_{int_i} < d_{ij} < R_{ext} \\ 0 & \text{if } d_{ij} \leq R_{int_i} \end{cases} \tag{22}$$

The modified linear velocity of the UGV_i is then given by:

$$\bar{v}_j = v_j \psi_i^j \tag{23}$$

Using the definition of R_{int_i} (where $R_{int_i} \neq R_{int_j}$), it is guaranteed that two UGVs do not stop simultaneously. Indeed, if the UGVs have the same R_{int_i} we can observe local minima in certain configurations, in fact, when $d_{ij} < R_{int_i}$ then $\psi_i^j = \psi_j^i = 0$ and the robots are stopped at the same time.vR_{ext} is designed according to communication constraints (latency) and localization errors (GPS). This penalty function can be straightforward integrated to our control architecture (cf. Fig. 2) by adding a block after the output of the *Control law* block (cf. Fig. 6).

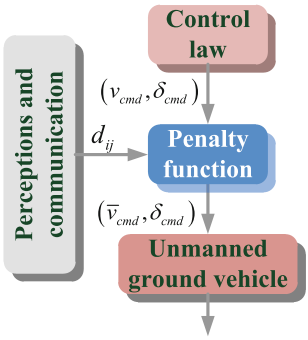


Fig. 6 Integration of the penalty function in the proposed architecture

4 Simulations

This section shows the navigation of a group of $N = 3$ UGVs in a cluttered environment using the proposed control architecture. The reconfiguration strategy (SFR) between the formation shapes is also analyzed. All simulations were made in MATLAB® software. The physical parameters of the used UGV are based on the urban vehicle VIPALAB from Apojee company [13]. The UGV constraints are minimum velocity $v_{min} = 0.1$ m/s, maximum velocity 1.5 m/s, maximum steering angle $\gamma_{max} = \pm 30°$ and maximum acceleration 1.0 m/s^2. We consider that the sample time is 0.01 s. Each UGV has a range sensor (LIDAR) with a maximum detected range equal to $D_{max} = 10$ m and a stable communication network.

The controller parameters are set to $\mathbf{K} = (1, 2.2, 8, 0.1, 0.01, 0.6)$ (cf. Sect. 2.2.3). These parameters were chosen to obtain a safe and smooth trajectory, fast response and velocity value within the limits of the vehicle capacities. The radius for non-collision between UGVs are selected as $R_{int_L} = 1.8$ m, $R_{int_1} = 2.2$ m, $R_{int_2} = 2.0$ m and $R_{ext} = 2.7$ m. For each simulation the vehicles start at the same configuration and must reach the same final configuration. The initial positions of the vehicles have an offset $(\Delta x, \Delta y) = (1,\ 0.5)$ m from the initial position of their assigned virtual targets.

The simulations given in Figs. 7, 8, 9, 10 and 11 are selected from several conclusive simulations because they make a focus on the proposed reconfiguration method

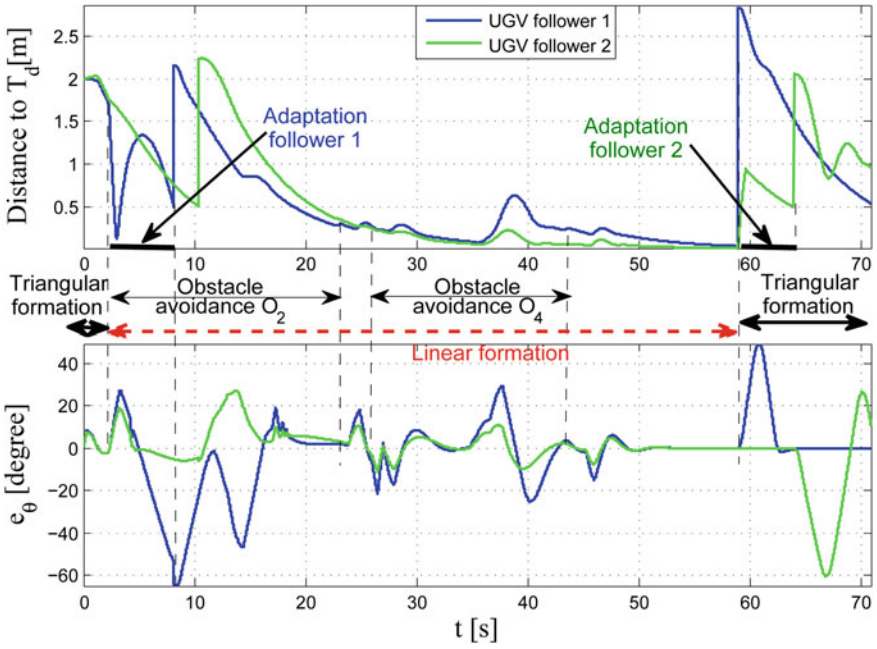


Fig. 7 Distance and orientation errors of the UGVs w.r.t. their virtual targets

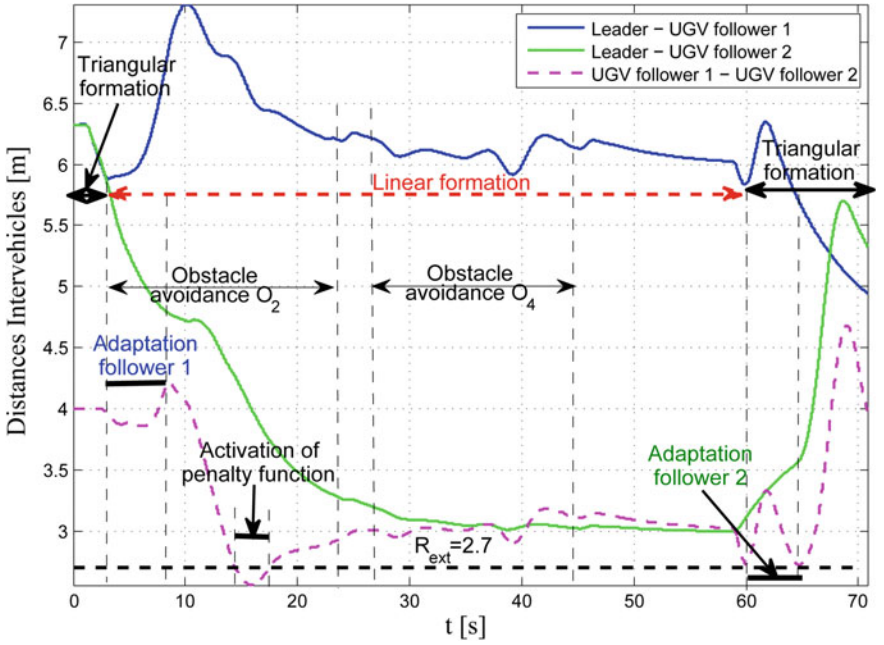


Fig. 8 Distance among the UGVs

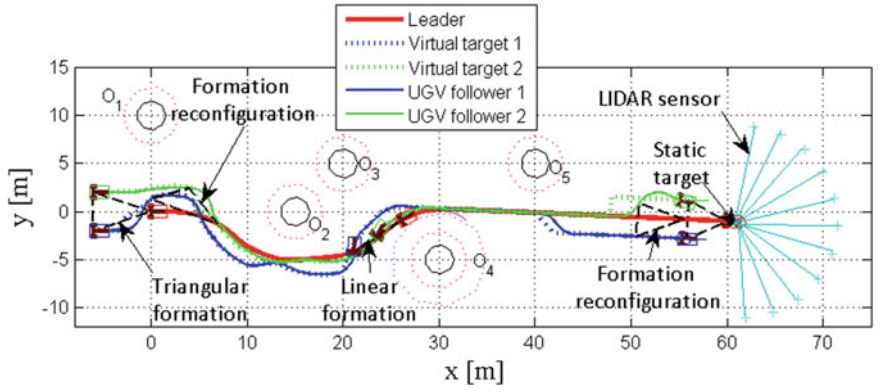


Fig. 9 Navigation with reconfiguration in formation for a group of $N = 3$ UGVs

between the two formation shape (triangular and linear shapes) while navigating in a cluttered environment (this simulation can be found online[1]). We consider that the initial formation coordinates are defined by $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2)$, with $\mathbf{f}_1 = (-4, -2)^T$ m and $\mathbf{f}_2 = (-4, 2)^T$ m (triangular shape). Therefore, the group of UGVs must keep the formation while moving in a cluttered environment. A static target is defined in the

[1] http://maccs.univ-bpclermont.fr/uploads/Profiles/VilcaJM/FormationReconfiguration.mp4.

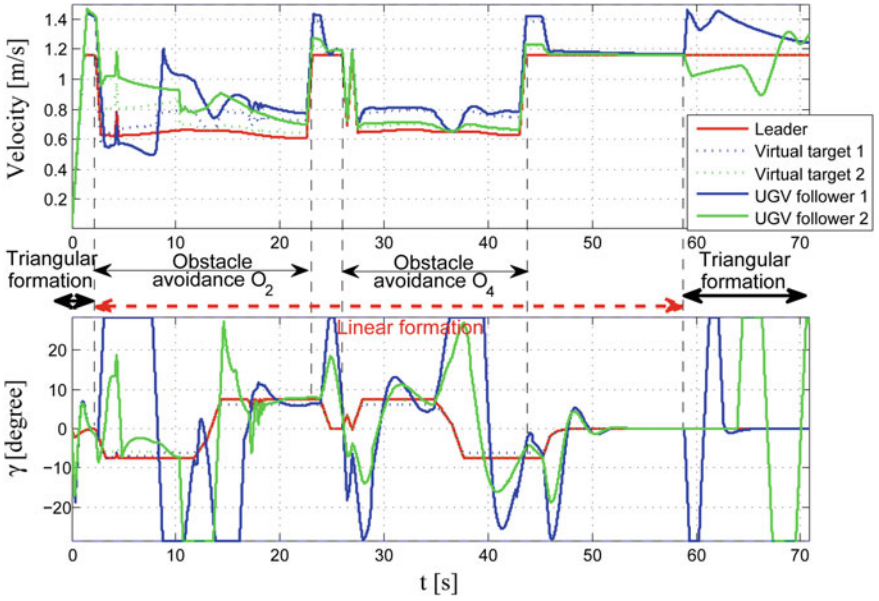


Fig. 10 Velocities commands of the UGVs

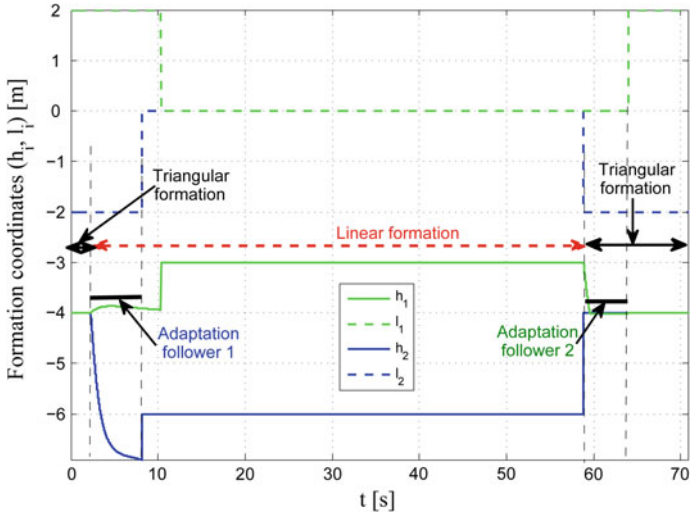


Fig. 11 Progress of the set-point definition $\mathbf{f}_i$ according to the proposed SFR

environment, the leader (and thus the formation) must go toward it while avoiding the hinder obstacle. The new targeted formation is defined as straight line with the following coordinates $\mathbf{F}^n = (\mathbf{f}_1^n, \mathbf{f}_2^n)$, with $\mathbf{f}_1^n = (-6, 0)^T$ m and $\mathbf{f}_2^n = (-3, 0)^T$ m.

At the beginning of the simulation (cf. Fig. 9), the navigation of the group of UGVs is in triangular formation $\mathbf{F}$. When the leader detects an obstacle with adequate range to allow the formation reconfiguration, then the leader avoids the obstacle using the limit-cycle method (limit-cycle is increased by $R_f = 2$ m to allow a safe navigation (cf. Sect. 2.2.2) and sends the new desired formation $\mathbf{F}^n$ to the other UGVs (followers) to modify the configuration of the formation. The formation returns to triangular shape $\mathbf{F}$, when the leader does not detect obstacles that can hinder the other UGVs movement and the last follower left behind the avoided obstacle. The adaptation phase allows to have the virtual target always ahead to the followers to obtain a suitable adaptive formation reconfiguration (cf. Figs. 7 and 11).

Figure 7 shows the values of errors d and e_θ between each UGV and its virtual target. At first reconfiguration, it can be observed that the follower 1 wait until its assigned virtual target is ahead (cf. Sect. 3.2.1). Moreover, it is noted some small peaks that are related to the fast dynamic change of the leader (the dynamic of the formation increased and the saturation occurs in the followers when the leader curvature increased). Figure 8 shows the distance between each UGV of the formation. This last figure shows clearly non-collision between the vehicles in the formation, i.e., $d_{ij} > R_{int_{12}}$ (cf. Sect. 3.2.2). The figures show some peaks which are related to the adaptation and reconfiguration phase between formations.

Figures 9 and 10 show respectively the trajectories and velocities of the UGVs. It can be noted that the vehicles trajectories are smooth along the navigation and there is not neither collisions with the obstacles nor inter-vehicle collisions. The reconfiguration strategy was designed to reduce the peaks of the control commands of each UGV when the transition between the formation occurs (cf. Fig. 10). The proposed control architecture allows thus to adapt the formation according to the environment context. Figure 11 shows the evolution of the formation coordinates (h_i, l_i) (virtual target positions). It can be observed that adaptation phase of h_i when the follower is always ahead of its new assigned virtual target (21) which attest on the efficiency of the strategy for formation reconfiguration.

5 Conclusions and Prospects

This paper presented an overall control architecture to cope with the navigation in formation of a group of UGVs in cluttered environment. A single *Control law* embedded in each UGV is used in the proposed architecture which allows the simplification of the overall control strategy for the navigation in formation. The obstacle avoidance based on the limit-cycle trajectories allows to keep the desired formation shape during the navigation even in cluttered environments. In the proposed formation definition based on *Leader-follower* approach, the leader reference path is not taken into account, only its current pose and dynamic has to be known by the follow-

ers. It allows thus more accurate and flexible formation navigation. A fully reactive reconfiguration strategy between the UGVs based on suitable smooth switching of the virtual target configurations was also proposed. This strategy avoids the use of predefined trajectories and it can be applied for different situations when the formation has to be modified according to the environment context (dynamic, cluttered, etc.). Furthermore, this strategy takes into account the probable collisions between vehicles as well as the vehicle constraints to ensure safe navigation to reach the new desired formation. Different accurate simulations using a tricycle vehicles show the efficiency and the flexibility of the proposed strategy for multi-robot navigation.

In future works, formation reconfiguration strategy even in uncertain environments (for instance, w.r.t. the vehicle's perception/localization) will be addressed.

Acknowledgments This work was supported by the French National Research Agency through the Safeplatoon project.

References

1. Adouane, L., Benzerrouk, A., Martinet, P.: Mobile robot navigation in cluttered environment using reactive elliptic trajectories. In: 18th IFAC World Congress (2011)
2. Benzerrouk, A., Adouane, L., Martinet, P.: Altruistic distributed target allocation for stable navigation in formation of multi-robot system. In: 10th International IFAC Symposium on Robot Control (SYROCO'12), Dubrovnik, Croatia (2012)
3. Chaimowicz, L., Kumar, R.V., Campos, M.F.M.: A mechanism for dynamic coordination of multiple robots. Auton. Robot. **17**, 7–21 (2004)
4. Chao, Z., Zhou, S.L., Ming, L., Zhang, W.G.: UAV formation flight based on nonlinear model predictive control. Math. Prob. Eng. **2012**, 1–15 (2012)
5. Chen, X., Li, Y.: Smooth formation navigation of multiple mobile robots for avoiding moving obstacles. Int. J. Control Autom. **4**(4), 466–479 (2006)
6. Clark, J., Fierro, R.: Mobile robotic sensors for perimeter detection and tracking. ISA Trans. **46**(1), 3–13 (2007)
7. Consolini, L., Morbidi, F., Prattichizzo, D., Tosques, M.: Leader-follower formation control of nonholonomic mobile robots with input constraints. Automatica **44**(5), 1343–1349 (2008)
8. Das, A., Fierro, R., Kumar, V., Ostrowski, J., Spletzer, J., Taylor, C.: A vision-based formation control framework. IEEE Trans. Robot. Autom. **18**(5), 813–825 (2002)
9. Dasgupta, P., Whipple, T., Cheng, K.: Effects of multi-robot team formations on distributed area coverage. Int. J. Swarm Intell. Res. **2**(1), 44–69 (2011)
10. Desai, J., Ostrowski, J., Kumar, V.: Modeling and control of formations of nonholonomic mobile robots. IEEE Trans. Robot. Autom. **17**(6), 905–908 (2001)
11. El-Zaher, M., Contet, J.M., Gechter, F., Koukam, A.: Echelon platoon organisation: a distributed approach based on 2-spring virtual links. In: 15th International Conference on Artificial Intelligence: methodology, Systems, Applications (AIMSA), Germany (2012)
12. Ghommam, J., Mehrjerdi, H., Saad, M., Mnif, F.: Formation path following control of unicycle-type mobile robots. Robot. Auton. Syst. **58**(5), 727–736 (2010)
13. The Institut Pascal Data Sets.: http://ipds.univ-bpclermont.fr (2013)
14. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**, 90–99 (1986)
15. Levinson, J., Thrun, S.: Robust vehicle localization in urban environments using probabilistic maps. In: IEEE International Conference on Robotics and Automation, Alaska, USA (2010)

16. Lozenguez, G., Adouane, L., Beynier, A., Mouaddib, A., Martinet, P.: Map partitioning to approximate an exploration strategy in mobile robotics. In: MAGS—Multiagent and Grid Systems (2012)
17. Luca, A.D., Oriolo, G., Samson, C.: Feedback control of a nonholonomic car-like robot. In: J.P. Laumond (ed.) Robot Motion Planning and Control, pp. 171–253. Springer-Verlag (1998)
18. Mesbahi, M., Hadaegh, F.: Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching. In: Proceedings of the 1999 IEEE International Conference on Control Applications, vol. 2, pp. 1211–1216 (1999)
19. Miner, D.: Swarm Robotics Algorithms: A Survey (2007)
20. Shames, I., Deghat, M., Anderson, B.: Safe formation control with obstacle avoidance. In: IFAC World Congress, Milan, Italy (2011)
21. Sinha, A., Ghose, D.: Generalization of linear cyclic pursuit with application to rendezvous of multiple autonomous agents. IEEE Trans. Autom. Control **51**, 1819–1824 (2006)
22. Tang, H., Song, A., Zhang, X.: Hybrid behavior coordination mechanism for navigation of reconnaissance robot. In: International Conference on Intelligent Robots and Systems (2006)
23. Vilca, J., Adouane, L., Mezouar, Y., Lébraly, P.: An overall control strategy based on target reaching for the navigation of an urban electric vehicle. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), Tokyo, Japan (2013)
24. Zapata, R., Cacitti, A., Lepinay, P.: Dvz-based collision avoidance control of non-holonomic mobile manipulators. JESA, Eur. J. Autom. Syst. **38**(5), 559–588 (2004)
25. Ze-su, C., Jie, Z., Jian, C.: Formation control and obstacle avoidance for multiple robots subject to wheel-slip. Int. J. Adv. Rob. Syst. **9**, 1–15 (2012)

A Graph-Based Formation Algorithm for Odor Plume Tracing

Jorge M. Soares, A. Pedro Aguiar, António M. Pascoal and Alcherio Martinoli

Abstract Odor plume tracing is a challenging robotics application, made difficult by the combination of the patchy characteristics of odor distribution and the slow response of the available sensors. This work proposes a graph-based formation control algorithm to coordinate a group of small robots equipped with odor sensors, with the goal of tracing an odor plume to its source. This approach makes it possible to organize the robots in arbitrary and evolving formation shapes with the aim of improving tracing performance. The algorithm was evaluated in a high-fidelity submicroscopic simulator, using different formations and achieving quick convergence and negligible distance overhead in laminar wind flows.

Keywords Odor source localization · Plume tracing · Formation control · Robotic olfaction

J.M. Soares (✉) · A. Martinoli
Distributed Intelligent Systems and Algorithms Laboratory, School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
e-mail: jorge.soares@epfl.ch

A. Martinoli
e-mail: alcherio.martinoli@epfl.ch

J.M. Soares · A.M. Pascoal
Institute for Systems and Robotics, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal

A.P. Aguiar
Department of Electrical and Computer Engineering, Faculty of Engineering, University of Porto, Porto, Portugal
e-mail: pedro.aguiar@fe.up.pt

A.M. Pascoal
National Institute of Oceanography, Dona Paula, Goa, India
e-mail: antonio@isr.ist.utl.pt

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_18

1 Introduction

As early as the 1950s, researchers were looking into the use of electronic devices for odor sensing, an application commonly referred to as *machine olfaction* [9, 24]. While it was not until several decades later that work in robotic olfaction began [27], it has been steadily intensifying ever since. The possibilities are endless, including very promising applications in search and rescue operations [11], as well as environmental and industrial safety [5].

Odor source localization is frequently divided into three stages: plume finding, plume traversal, and source declaration. We concentrate on the intermediate and most frequently studied, plume traversal or tracing, i.e., following a chemical plume to its source. Odor plume tracing presents particular challenges when compared to other search problems: due to turbulent transport, chemical concentration in a plume tends to be very patchy, with packets of high concentration and periods of low or completely absent odor [26]. Along with the relatively slow response of odor sensors, this makes it hard or impractical to use simple gradient ascent techniques.

Previous work in the field has resulted in a significant number of methods for odor tracing, of which a detailed survey is provided in [14]. Many algorithms take inspiration from living creatures, such as bacteria [4] or silkworm moths [20], and generally operate by switching a single robot among a set of simple behaviors. Experiments have also been conducted with multiple robots [10, 13, 17], acting both independently and cooperatively. Other approaches include Braitenberg-type control [16], probabilistic inference [15, 17, 28] and meta-heuristic optimization methods [1–3, 12, 21].

We choose to focus on formation-based algorithms, which take advantage of multiple cooperating robots moving in formation to locate the source with minimal wandering. Our work is strongly inspired by the crosswind formation work in [18], as well as other formation- and swarm-based approaches [8, 10, 19]. This decision is driven by the choice of success metric: while the energy budget goes up with the number of robots, these techniques allow us to minimize the distance overhead and time to the source, a requirement in scenarios such as the aforementioned search and rescue operations. Furthermore, our solution requires very little processing power, is easy to implement, and requires no sensor information other than wind, odor and relative positions—wheel encoder information can optionally be used to improve the behavior of the algorithm.

A graph-based formation controller drives the robots to an arbitrary leaderless formation. It is based on the principle of Laplacian feedback, which has been used extensively in other contexts [6, 22]. The only information required by each robot is the relative position of its neighbors, and the formation may take an arbitrary shape and dynamically change over time. Each robot broadcasts its own odor reading, which is used to adjust the separation between robots and to bias the movement of the formation so as to center it over the plume. At all times, robots are configured with an urge to move upwind, i.e., in the expected direction of the plume source. The

Fig. 1 *Left* Fully equipped Khepera III robot, including the range and bearing board (*bottom*), odor sensing board (*middle*) and wind sensing board (*top*). *Right* Inside view of the DISAL wind tunnel

details of the algorithm are presented in Sect. 2. Note that the core of this paper is not a specific formation proposal, but a framework that allows us to use any desired formation, possibly shaped by the underlying odor plume.

The proposed solution was validated and evaluated in a submicroscopic simulator that mimics our existing real-world infrastructure, including the Khepera III differential wheeled robots and add-on odor and wind sensing boards. The simulation arena is a model of the DISAL wind tunnel, using laminar wind flow and a filament-based odor propagation model. The fully equipped Khepera III robot and an inside view of the wind tunnel are shown in Fig. 1, while Sect. 3 details the simulations and results obtained for two example formation shapes.

2 Controller Design

Wind takes a central role in our algorithm, as odor displacement (and desired robot displacement) mostly takes place along the wind direction. Furthermore, the lack of a heading sensor renders wind our only source for global orientation.

The formation shape is specified in terms of robot positions relative to the formation center, in an *upwind–crosswind* frame, coupled with the wind direction. This allows the entire formation to rotate and align to the wind. Throughout this paper, we use the terms *forward* and *backward* to refer to the nodes respectively upwind and downwind in the desired formation. The definition of *left* and *right* of center follows naturally. The reference formation shape, as well as the relative position of each robot in it, are defined a priori and known by all robots—therefore, robots are able to independently determine which should be treated as left or forward of center.

The controller consists of a minimal set of independent behaviors: formation keeping, which drives and maintains the robots in the desired formation, upwind movement, which moves robots upwind to the source, and plume centering, which keeps the overall formation centered on the plume. Each behavior outputs a desired

velocity vector and these vectors are then combined and transformed into a control signal.

Control is distributed and asynchronous, and no supervisory information or intervention is assumed; moreover, there is no absolute positioning. Consequently, the controller operates on a local frame, in which the y axis is aligned with the robot front, pointing forward, and the x axis points 90° anticlockwise. Given the short inter-distances involved, we consider that all robots are within communication range. Nevertheless, data exchange is minimized, and the only information explicitly exchanged among the robots is the most current odor measurement.

Each behavior is described in detail in the subsequent sections. As control laws are formulated locally and homogeneous across all robots, the subscript self-identifying the node is omitted except where otherwise noted.

2.1 Sensing

Wind and odor are assumed to be measured by the sensors introduced in [17], where they are fully described. No extensive discussion of the sensor characteristics will be done in this section as experiments took place using simulation equivalents for which the errors models are provided in the evaluation section.

Wind measurements are provided by a custom board featuring six NTC thermistors inside individual 3D printed tubes arranged in a star shape. As wind measurements are affected by significant noise, a discrete scalar Kalman filter is used to obtain an estimate of the relative wind direction. The wind observations are complemented by odometry information from the wheel encoders, compensating for the rotation of the robot. We consider the wind angle θ to be zero when the robot is facing the wind.

For the odor measurements, a MiCS-5521 VOC sensor is used, with a gas pump ensuring continuous flow. The sensor provides noisy quantized measurements, in the 0–1000 range. The odor readings are broadcast by each robot, along with its identification. Each robot stores the latest concentration received from every team mate, c_j, and updates three values used in the algorithm to influence the geometry and movement of the formation:

- c_c, the mean of concentrations measured by the center robots
- c_l, the mean of concentrations measured by robots left of the formation center
- c_r, the mean of concentrations measured by robots right of the formation center

Finally, relative positions between the robots are provided by an extension board equipped with sixteen LEDs, allowing the robot to obtain the range, bearing and ID of each neighbor in sight [25].

2.2 Laplacian Feedback

Our formation can be expressed as an undirected graph $G = (V, E)$, in which vertices V correspond to robots and edges E correspond to inter-robot relative positioning links, or a subset thereof. From there, we can use the work in [22] and standard results in graph theory to attain a provably stable solution to the formation control problem:

$$\dot{\mathbf{x}} = -\left(\mathscr{L} \otimes I_2\right)(\mathbf{x} - \mathbf{b}) \tag{1}$$

where $\mathscr{L} = \mathscr{B}\mathscr{B}^T$ is the positive-semidefinite Laplacian matrix, obtained from the incidence matrix $\mathscr{B}$ that describes the edges of G. The (x, y) absolute position vector for all robots is given by $\mathbf{x}$, and the desired offsets to the formation centroid are given by the bias vector $\mathbf{b}$. As stated above, the law is only applicable under the assumptions of holonomicity, absolute positioning, and full connectivity.

The same approach can, however, be implemented in a decentralized fashion and using only relative positioning information, assuming a connected but not necessarily complete graph, and accounting for nonholonomicity either natively or by offloading to a lower level controller [6]. As previously discussed, we want the formation to be oriented with respect to the wind, requiring the rotation of the bias vectors by the measured wind speed θ. The resulting velocity vector for formation control is

$$\mathbf{u_f} = -\begin{bmatrix} \sum_{j=0}^{N} \mathscr{L}_j \left(x_j - \beta_j^x\right) \\ \sum_{j=0}^{N} \mathscr{L}_j \left(y_j - \beta_j^y\right) \end{bmatrix} \tag{2}$$

where $\mathscr{L}_j = \mathscr{L}_{i,j}$ is the entry of the Laplacian matrix relating the controlled node i to neighbor j, x_j and y_j are the relative positions to robot j in the body frame, and β_j is a local analogue to $\mathbf{b}$, describing the desired relative position between the two robots in the robot frame, i.e. $\beta_j = R(\theta_i)[\bar{\mathbf{p}}_\mathbf{j} - \bar{\mathbf{p}}_\mathbf{i}]$, $\bar{\mathbf{p}}_\mathbf{i}$ and $\bar{\mathbf{p}}_\mathbf{j}$ expressed in the wind/formation frame. We do not consider role assignment in this work, and so the position of each robot in the formation is defined by its identifier.

2.3 Dynamic Spacing

Depending on the size and growth rate of the odor plume, a fixed formation might not be an optimal choice. Therefore, we implement a method to change the formation spacing by varying the bias vector as a function of the measurements.

For simplicity, we define two scalar parameters, s_{cw} and s_{uw}, which represent adaptive bias coefficients in the crosswind and upwind direction, respectively. Assuming

biases (and, hence, the formation) are symmetric, this results in a minor modification. Equation (2) remains valid, but β_j assumes a new formulation:

$$\beta_j = R(\theta_i) \begin{bmatrix} s_{cw} & 0 \\ 0 & s_{uw} \end{bmatrix} [\bar{\mathbf{p}}_\mathbf{j} - \bar{\mathbf{p}}_\mathbf{i}] \tag{3}$$

In the remainder of the paper, we use a constant upwind scaling factor and continuously vary the crosswind scaling according to Eq. (4). The underlying rationale is that, at our evaluation scale, the differences in the plume structure are more pronounced in the crosswind direction. The scaling methods may, however, be easily modified to adapt to different realities.

$$s_{cw} = k_{cw} \frac{c_l + c_r}{1 + c_c} \tag{4}$$

Proper limits need to be set to account for the minimum safe distance and the maximum communication/movement range. We have tuned k_{cw} to approximately maintain the side robots on the detectable edges of the plume, for both the considered line and rectangular formations; values between 0.5 and 1.0 appear to yield reasonable results. However, the distribution of robots in the plume depends on its aspect ratio, strongly influenced by wind speed and other factors mediating odor dispersal.

2.4 Upwind Movement and Centering

While in the plume, robots should move upwind in the presumed direction of the source. Therefore, a movement urge in the direction of the apparent wind is defined as

$$\mathbf{u_w} = R(\theta) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5}$$

We want to keep the formation centered in the plume, which is achieved by adding a crosswind force depending on the difference between c_l and c_r, the aforementioned means of the odor readings to the left and right of the formation center. To prevent extreme variations in control outputs due to the wide amplitude of odor measurement variations, we implement a logistic response given by

$$\mathbf{u_c} = R(\theta) \begin{bmatrix} \frac{1}{1+e^{-(c_l - c_r)/k_l}} - 0.5 \\ 0 \end{bmatrix} \tag{6}$$

The value of k_l should be approximately of the same dimension as the range of the sensor. In our experiments, $k_l = 200$. This asymptotically limits the maximum

requested crosswind velocity to ± 0.5, a value that is only approached for highly asymmetric odor readings.

2.5 *Behavior Aggregation*

We combine the requested velocity vectors of each behavior with a weighted sum

$$\mathbf{u} = k_w \mathbf{u_w} + k_c \mathbf{u_c} + k_f \mathbf{u_f} \tag{7}$$

At this stage, constant weights $k_w = k_c = k_f = 1$ are used. For the particular functions described above, this yields

$$\mathbf{u} = -\begin{bmatrix} \sum_{j=0}^{N} \mathscr{L}_j \left(x_j - \beta_j^x \right) \\ \sum_{j=0}^{N} \mathscr{L}_j \left(y_j - \beta_j^y \right) \end{bmatrix} + R(\theta) \begin{bmatrix} \frac{1}{1+e^{-(c_l - c_r)/k_l}} - 0.5 \\ 1 \end{bmatrix} \tag{8}$$

where β_j is given by (3). The resulting vector $\mathbf{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$ is then used to determine the requested (dimensionless) linear and angular velocities using simple proportional controllers (9) and (10), limited to forward movement and saturated at a reasonable maximum within the operating range. These are passed to a motion controller that computes the actual control signals.

$$v = k_v \mathbf{u}_y \qquad 0 \leq v \leq v_{max} \tag{9}$$

$$\omega = k_\omega \mathbf{u}_x \qquad -\omega_{max} \leq \omega \leq \omega_{max} \tag{10}$$

3 Evaluation

To validate and evaluate the performance of our solution, we use a simulated equivalent of our real world wind tunnel, robots and related sensors. It is built around Webots [23], a submicroscopic robotics simulator, configured and calibrated to approximate real world behavior [17].

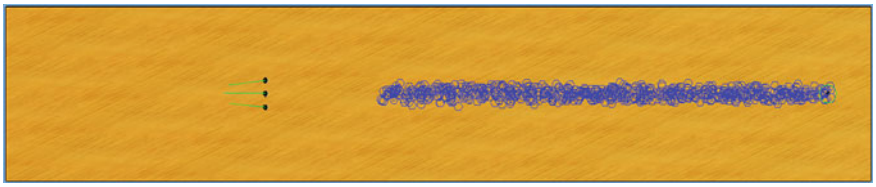


Fig. 2 Simulation setup, with the source on the *right* and robots starting on the *left*. The plume is shown in *blue*, and consists of filaments with growth rate $\gamma = 4 \times 10^{-7}\ \mathrm{m^2\ s^{-1}}$. The wind flows *right*-to-*left*. For ease of visualization, the filaments are drawn enlarged

3.1 Setup

The simulation arena, shown in Fig. 2, is a 20 m $\times$ 4 m plane. The robots start approximately 14 m downwind (depending on the exact starting formation), and the source is placed at the 1 m mark, centered.

Odor propagation is modeled using filaments and the wind flow is constant. We use the built-in model of the Khepera III, and the wind, odor and range and bearing mechanisms are based on those available to the real robots [18]. Details of the wind and odor simulations are provided in the following subsections. Relative positioning, in the form of range and bearing measurements, is provided to the robots with angular error of 0.1 rad and range error of 10 % of the distance, and is locally converted to $x - y$ coordinates.

Our simulation environment does not presently handle occlusion and always provides the range and bearing to every other robot. This is a significant limitation in the case of line formations, where occlusions are the norm. We compensate by designing the graph so that a robot only considers its two nearest neighbors in the desired formation (or just the nearest, for robots at the ends). As measurements are noisy, considering fewer neighbors leads to a less stable formation, and so our simulation underestimates expected real-world formation control performance. It also hinders the built-in collision avoidance mechanism, a problem not addressed in this work. Source declaration is outside the scope of this paper, so simulations are terminated by an external supervisor when the robots are in the vicinity of the source.

No attempt was made to maximize the robot movement speed and, consequently, minimize the mission completion time. Instead, the target speed is defined by the low-level translation of $|u_w| = 1$ into wheel speeds, and is of approximately 7 cm $\mathrm{s^{-1}}$. The simulation step is 32 ms. Communication is assumed perfect, although moderate packet loss can be tolerated with no substantial impact.

3.1.1 Wind Simulation and Sensing

A constant wind field of 1 m $\mathrm{s^{-1}}$ was used, simulating laminar flow. The wind sensor provides noisy wind velocity measurements, resulting of the sum of the (x, y, z)

wind velocity at the position of the robot (w) with a vector of three independent Gaussian random variables, $N(0, \sigma_a^2)$.

As the wind field is constant, the magnitude of w is also constant and only its orientation changes. The standard deviation of the Gaussian noise is set to $\sigma_a = 0.1\ \mathrm{m\ s^{-1}}$. This differs from the noise distribution of the real sensor, for which the experimentally determined distribution of the *direction* noise is $N(0, \sigma_d^2)$, $\sigma_d = 4°$.

3.1.2 Odor Simulation and Sensing

The odor propagation implementation is based on the the filament-based model proposed by Farrell et al. [7], and generates an intermittent plume similar to the one observed in the wind tunnel. Odor is simulated as a set of filaments ($i = 0, ..., N$), each containing a constant amount $s = 8.3 \times 10^9$ of molecules. Each filament is defined by its (x, y, z) position $\mathbf{p}_{i,t}$, and its width $w_{i,t}$.

At each time step, the position of a filament is updated according to the wind flow and a stochastic process $\mathbf{v}_p$, consisting of a vector of three independent Gaussian random variables, $N(0, \sigma_p^2)$, with $\sigma_p = 0.1$ m. Molecular dispersion is modeled by having the filament width increase with time while the peak concentration decreases. The resulting evolution of the filaments is described by

$$\mathbf{p}_{i,t+\Delta t} = \mathbf{p}_{i,t} + \mathbf{a}(\mathbf{p}_{i,t})\Delta t + \mathbf{v}_p \tag{11}$$

$$w_{i,t+\Delta t} = w_{i,t} + \frac{\gamma}{2w_{i,t}} \tag{12}$$

The filament dispersion rate approximating the wind tunnel conditions was previously determined to be $\gamma = 4 \times 10^{-7} m^2\ s^{-1}$. The virtual odor source releases 100 filaments per second with an initial width of $w_{i,0} = 10$ cm and initial position distributed over the circular area of the source. The odor concentration at time t and position $\mathbf{p}$ is the sum of the concentration contribution of all filaments, which decays exponentially with the increasing distance to the center of a filament, that is,

$$c_t(\mathbf{p}) = \sum_{i=0}^{N} \frac{s}{w_{i,t}^3} \exp\left(-\frac{|\mathbf{p} - \mathbf{p}_{i,t}|}{w_{i,t}^2}\right) \tag{13}$$

No noise is added to the measured concentration as the chemical-to-electrical transduction noisy was observed to be negligible on the real platform [17]. The samples are run through a sliding window filter, which outputs the highest amongst the 50 most recent readings; in our simulation, this corresponds to a window of 1.6 s. This serves the dual purpose of emulating the approximately 1 s recovery time of the physical sensor and of adding additional noise filtering.

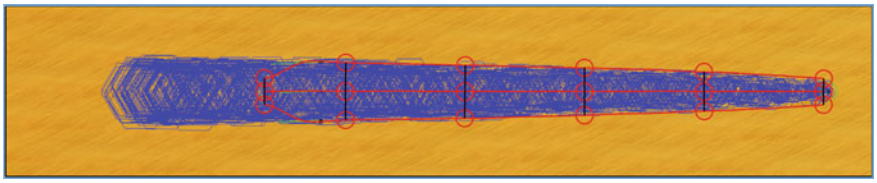


Fig. 3 Robot trajectories for a three-robot line formation in a plume with filament growth rate $\gamma = 10^{-3}\text{m}^2\ \text{s}^{-1}$. *Black lines* connect the robot positions at intervals of approximately 35 s

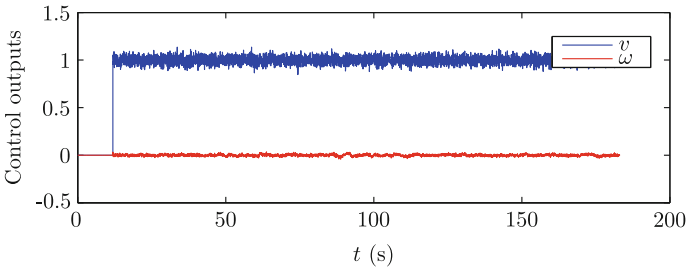


Fig. 4 Control outputs v and ω for the center robot. Both are dimensionless quantities, converted by a low level controller into the actual linear and angular speeds. The robots were parked waiting for the first nonzero odor measurements during the first 11 s

3.2 Results

Experiments were run for multiple formation shapes in distinct plume conditions. We provide the results for two illustrative formations: a three-robot line formation, and a five-robot rectangular formation. The results presented here are the product of single simulations, but representative of what was observed in multiple runs.

3.2.1 Line Formation

The first simulation was run using a three-robot linear formation, oriented along the crosswind axis, corresponding to predefined biases $\mathbf{b}_{upwind} = \begin{bmatrix}0\ 0\ 0\end{bmatrix}^T$ and $\mathbf{b}_{crosswind} = \begin{bmatrix}1\ 0\ -1\end{bmatrix}^T$. This simulation uses a high filament growth rate $\gamma = 10^{-3}\text{m}^2\ \text{s}^{-1}$ to better highlight the adaptive formation spacing. For the same reason, the robots start centered on the plume, in close proximity—this is not a requirement, and the next section presents simulations with off-center starting positions.

Figure 3 shows the trajectories negotiated by the three robots over a total time of 180 s, overlapped with a snapshot of the simulation environment at an arbitrary time. According to the odor concentrations measured, robots begin by widening the formation and then successfully trace the limits of the plume to its source. At this scale, the trajectories appear smooth, with no major disturbances.

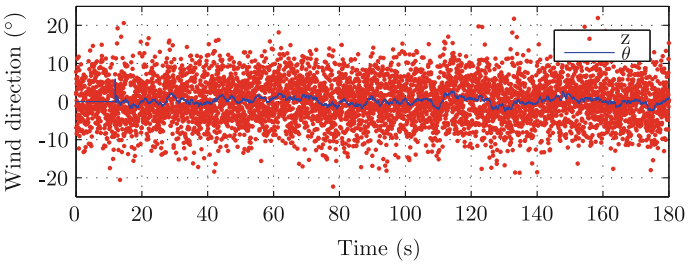


Fig. 5 Wind direction measurements z and Kalman estimate θ, relative to the front of the center robot

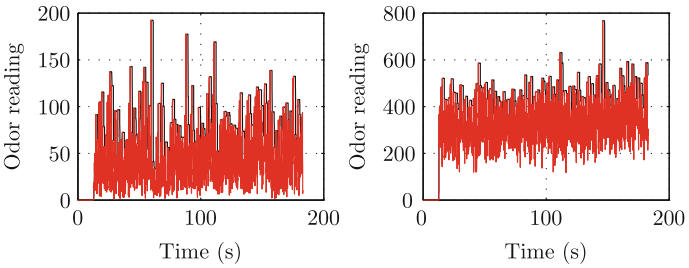


Fig. 6 Odor measurements for robots 1 (*left*) and 2 (*center*). The *red lines* show the instant odor measurements, and the *black lines* a sliding window maximum filtering. Note the different scales

In fact, looking at the control signals in Fig. 4, we can see that the controller is very stable in the angular speed ω, but less so in the linear speed v, which shows a standard deviation $\sigma = 0.0762$.

Closer analysis of the results shows that, in spite of the noisy wind and odor measurements, the dominating noise present in the control outputs is introduced by the formation control, a consequence of errors in the relative positions received from the range and bearing module. Improvements could be realized by introducing a Bayesian filter for neighbor tracking, and even further by broadcasting the control outputs and wind direction estimate of each vehicle and using them to better predict future relative positions. However, the fast rate of the sensor allows us to obtain good results even in the absence of more involved strategies.

Given the central role the wind direction takes in our control law, measurement error can have a significant impact on the performance. Figure 5 shows the wind direction measurements along the complete trajectory, as well as the Kalman filter estimate of the true wind direction, a considerable improvement over the raw data.

Finally, we illustrate our statements about the intermittency of the odor plume with the recorded odor measurements from robots 1 (left) and 2 (center) in Fig. 6. While the center robot reports higher odor readings, both series show high frequency and high amplitude variation. The 50-slot sliding window maximum filter helps obtain more relevant readings, but even its output is highly noisy. Nevertheless, the logistic

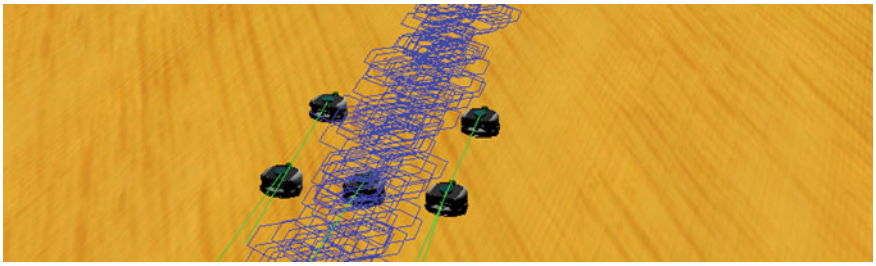


Fig. 7 Square formation. The *green lines* over the robots represent instant wind velocity measurements

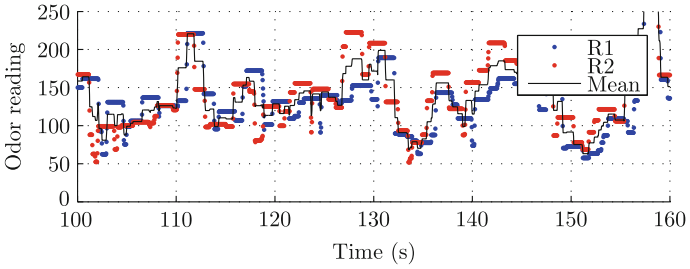


Fig. 8 Max-filtered odor measurements for robots 1 (*left*, *back*) and 2 (*left*, *front*), and resulting mean

response in centering and relatively limited changes in formation spacing are able to minimize the impact of these variations.

3.2.2 Rectangular Formation

The rectangular formation is composed of five robots: four on the vertices and one in the center of the rear edge. This corresponds to predefined biases $\mathbf{b}_{upwind} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$ and $\mathbf{b}_{crosswind} = \begin{bmatrix} 1 & 1 & 0 & -1 & -1 \end{bmatrix}^T$. An image of the formation is provided in Fig. 7. The robots start in two clusters on the sides of the tunnel, to showcase the plume centering and formation control capabilities. Each robot is connected to the two closest neighbors along the perimeter of the formation.

In contrast to the line formation, this one introduces spatial diversity in the upwind direction. Robots along the same upwind line are able to average their readings, in order to provide the aggregate measurement c_l and c_r, less affected by individual odor packets. Figure 8 shows a 60 s plot of the readings obtained by the two robots on the left edge of the rectangle and the resulting mean value, used as input to the controller. An additional benefit of upwind diversity, unexplored in this paper, is the possibility of using comparatively simple formation-based source declaration mechanisms.

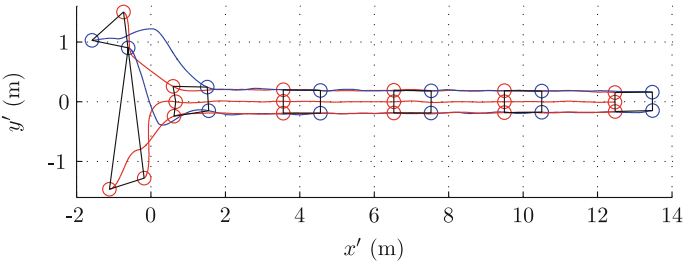


Fig. 9 Robot trajectories for a square formation in a plume with growth rate $\gamma = 4 \times 10^{-7} m^2\ s^{-1}$. The trajectories of the forward robots are traced in *blue*, and those of the backward robots are traced in *red*. *Black lines* connect the robot positions at intervals of approximately 35 s

The trajectories followed by the robots are presented in Fig. 9. The robots converge to the desired formation and plume center in the first 30 s, and continue upwind along the target trajectory. This simulation was run with the standard filament growth rate, therefore no change in plume width is observable in this short distance.

4 Conclusions

This paper presented a formation-based controller for odor plume tracing that uses multiple cooperating robots to trace a chemical plume to its source in laminar flow conditions. Formation control is based on Laplacian feedback and is capable of stabilizing arbitrary shapes. The formation geometry is adjusted over time to better envelope the plume and its motion combines upwind movement and crosswind alignment.

Simulations were performed to validate the performance of the algorithm, using both a three-robot linear formation and a five-robot rectangular formation. The experiments show that the algorithm can cope with noisy readings of wind direction, odor intensity and relative positions and that, after converging to the desired positions, the robots move along the centerline of the plume, tracing it to the source.

In the future, we plan to move from simulation to real-world experiments in our wind tunnel, possibly introducing Bayesian filtering on the relative positions. Afterwards, we will continue this work by further exploring the impact of different formation shapes and wind conditions through systematic experimentation.

Acknowledgments This work was partially funded by project PEst-OE/EEI/LA0009/2013 and grant SFRH/BD/51073/2010 from Fundação para a Ciência e Tecnologia. We sincerely thank Ali Marjovi at DISAL for the detailed and constructive comments.

References

1. Cabrita, G., Marques, L., Gazi, V.: Virtual cancelation plume for multiple odor source localization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5552–5558 (2013). doi:10.1109/IROS.2013.6697161
2. Cao, M.L., Meng, Q.H., Wang, X.W., Luo, B., Zeng, M., Li, W.: Localization of multiple odor sources via selective olfaction and adapted ant colony optimization algorithm. In: IEEE International Conference on Robotics and Biomimetics, pp. 1222–1227 (2013). doi:10.1109/ROBIO.2013.6739631
3. de Croon, G., O'Connor, L., Nicol, C., Izzo, D.: Evolutionary robotics approach to odor source localization. Neurocomputing **121**, 481–497 (2013). doi:10.1016/j.neucom.2013.05.028
4. Dhariwal, A., Sukhatme, G., Requicha, A.: Bacterium-inspired robots for environmental monitoring. In: IEEE International Conference on Robotics and Automation, pp. 1436–1443 (2004). doi:10.1109/ROBOT.2004.1308026
5. Distante, C., Indiveri, G., Reina, G.: An application of mobile robotics for olfactory monitoring of hazardous industrial sites. Ind. Rob. Int. J. **36**(1), 51–59 (2009). doi:10.1108/01439910910924675
6. Falconi, R., Gowal, S., Martinoli, A.: Graph based distributed control of non-holonomic vehicles endowed with local positioning information engaged in escorting missions. In: IEEE International Conference on Robotics and Automation, pp. 3207–3214 (2010). doi:10.1109/ROBOT.2010.5509139
7. Farrell, J.A., Murlis, J., Long, X., Li, W., Cardé, R.T.: Filament-based atmospheric dispersion model to achieve short time-scale structure of odor plumes. Environ. Fluid Mech. **2**(1–2), 143–169 (2002). doi:10.1023/A:1016283702837
8. Genovese, V., Dario, P., Magni, R., Odetti, L.: Self organizing behavior and swarm intelligence in a pack of mobile miniature robots in search of pollutants. IEEE/RSJ Int. Conf. Intell. Rob. Syst. **3**, 1575–1582 (1992). doi:10.1109/IROS.1992.594225
9. Hartman, J.: A possible method for the rapid estimation of flavours in vegetables. Proc. Am. Soc. Hort. Sci. **64**, 335–342 (1954)
10. Hayes, A., Martinoli, A., Goodman, R.: Distributed odor source localization. IEEE Sens. J. **2**(3), 260–271 (2002). doi:10.1109/JSEN.2002.800682
11. Ishida, H., Nakamoto, T., Moriizumi, T., Kikas, T., Janata, J.: Plume-tracking robots: a new application of chemical sensors. Biol. Bull. **200**(2), 222–226 (2001)
12. Jatmiko, W., Sekiyama, K., Fukuda, T.: A PSO-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement. IEEE Comput. Intell. Mag. **2**(2), 37–51 (2007). doi:10.1109/MCI.2007.353419
13. Khalili, A., Rastegarnia, A., Islam, M.K., Yang, Z.: A bio-inspired cooperative algorithm for distributed source localization with mobile nodes. In: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 3515–3518 (2013). doi:10.1109/EMBC.2013.6610300
14. Kowadlo, G., Russell, R.A.: Robot odor localization: a taxonomy and survey. Int. J. Robot. Res. **27**(8), 869–894 (2008). doi:10.1177/0278364908095118
15. Li, J.G., Meng, Q.H., Wang, Y., Zeng, M.: Odor source localization using a mobile robot in outdoor airflow environments with a particle filter algorithm. Auton. Rob. **30**(3), 281–292 (2011). doi:10.1007/s10514-011-9219-2
16. Lilienthal, A., Duckett, T.: Experimental analysis of gas-sensitive Braitenberg vehicles. Adv. Robot. **18**(8), 817–834 (2004). doi:10.1163/1568553041738103
17. Lochmatter, T.: Bio-inspired and probabilistic algorithms for distributed odor source localization using mobile robots. Ph.D. thesis 4628, EPFL (2010). doi:10.5075/epfl-thesis-4628
18. Lochmatter, T., Göl, E., Navarro, I., Martinoli, A.: A plume tracking algorithm based on crosswind formations. In: International Symposium on Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics (2013), vol. 83, pp. 91–102 (2010). doi:10.1007/978-3-642-32723-0_7

19. Marjovi, A., Marques, L.: Optimal swarm formation for odor plume finding. IEEE Trans. Cybern. **99** (2014). doi:10.1109/TCYB.2014.2306291
20. Marques, L., Nunes, U., de Almeida, A.T.: Olfaction-based mobile robot navigation. Thin Solid Films **418**(1), 51–58 (2002). doi:10.1016/S0040-6090(02)00593-X
21. Marques, L., Nunes, U., Almeida, A.T.: Particle swarm-based olfactory guided search. Auton. Rob. **20**(3), 277–287 (2006). doi:10.1007/s10514-006-7567-0
22. Mesbahi, M., Egerstedt, M.: Graph Theoretic Methods in Multiagent Networks. Princeton University Press, Princeton (2010)
23. Michel, O.: Webots: professional mobile robot simulation. Int. J. Adv. Rob. Syst. **1**(1), 39–42 (2004). doi:10.5772/5618
24. Moncrieff, R.W.: An instrument for measuring and classifying odors. J. Appl. Physiol. **16**(4), 742–749 (1961)
25. Pugh, J., Raemy, X., Favre, C., Falconi, R., Martinoli, A.: A Fast onboard relative positioning module for multirobot systems. IEEE/ASME Trans. Mechatron. **14**(2), 151–162 (2009). doi:10.1109/TMECH.2008.2011810
26. Roberts, P.J.W., Webster, D.R.: Turbulent Diffusion. ASCE Press, Reston, Virginia (2002)
27. Rozas, R., Morales, J., Vega, D.: Artificial smell detection for robotic navigation. In: International Conference on Advanced Robotics, pp. 1730–1733 (1991). doi:10.1109/ICAR.1991.240354
28. Vergassola, M., Villermaux, E., Shraiman, B.I.: 'Infotaxis' as a strategy for searching without gradients. Nature **445**(7126), 406–409 (2007). doi:10.1038/nature05464

Multi-agent Visibility-Based Target Tracking Game

Mengzhe Zhang and Sourabh Bhattacharya

Abstract In this paper, we address the problem of visibility-based target tracking for a team of mobile observers trying to track a team of mobile targets. Based on the results of previous work, the notion of *pursuit fields* around a single corner is introduced. We use the pursuit fields to generate navigation strategies for a single observer to track a single target in general environments. In order to tackle the case when more than one observer or target is present in the environment, we propose a two level hierarchical approach. At the upper level, the team of observers use a ranking and aggregation technique for allocating each target to an observer. At the lower level, each observer computes its navigation strategy based on the results of the single observer-single target problem, thereby, decomposing a large multi-agent problem into several 2-agent problems. Finally, we present a scalable algorithm that can accommodate an arbitrary number of observers and targets. The performance of this algorithm is evaluated based on simulation and implementation.

Keywords Multi-agent · Target tracking · Pursuit-evasion game

1 Introduction

Surveillance can be described as the monitoring of behavior, activities of other changing information. It includes many application areas such as crime prevention, wildlife monitoring, traffic monitoring and industrial processes. Government and law enforcement utilize relevant technologies on maintaining social control, recognizing and monitoring threats, and preventing criminal activities. Target tracking is a special class of surveillance in which the objective is to track the current state of a dynamic entity or an object of interest. Traditionally, sonar, radio [1] or infrared sensors [2] are used for target tracking tasks. With the emergence of computer vision, visibility-

M. Zhang · S. Bhattacharya (✉)
Iowa State University, Ames, USA
e-mail: sbhattac@iastate.edu

M. Zhang
e-mail: mengzhez@iastate.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_19

based target tracking received a lot of interest in the research community. In [3], the problem of maintaining visibility of a moving target is introduced. Authors propose algorithms for both predictable and unpredictable target, based on the received information of target's future actions. In our work, we consider the latter case. A centralized algorithm is proposed for a team of observers to track a team of targets. In [4], a distributed control algorithm is suggested which utilizes the information from vision sensors and communication. The algorithm also contains a mechanism which can predict the minimum time before an observer loses its target. In [5], vision-based object detection and tracking techniques are studied in depth for underwater robots.

There has been a considerable amount of research to address the single observer-single target tracking problem. We only mention a few of them here. You can refer to [6, 7] for an extensive review of previous work in target tracking. In [8], authors introduce a notion of escape risk and generates the escape-path tree for computing motion strategy of the observer. The escape-path tree can be obtained by taking local measurements and storing the most effective escape routes for a target to escape observer's field of view. In [9–11], stealth tracking problem is analyzed using vision sensors. Risk function is formulated so that observer can determine its moving direction. In [12], authors tackle the tracking task of a single evader in a dynamic environment. The approach is carried out based on three separate components, namely, occlusion advisor, collision advisor and decision maker. In [13], authors present a stochastic approach to maintain a nominal distance between an unmanned aerial vehicle and a ground-based moving target. The aforementioned works are different from ours because they only consider the scenario of single observer and single target.

Recently, there has been some effort to address the problem when a team of observers is deployed for the tracking task. In [14], authors utilize the local force vectors that cause robots to be attracted to nearby targets, and to be repelled from nearby robots. A distributed heuristic approach is proposed based on the force vectors. In [15], a new task assignment algorithm is presented which integrates area search and target tracking. In [16–18], authors use a coarse deployment controller and a target-following controller to control the robot deployment based on a region-based approach. They present an algorithm that treats the densities of robots and targets as properties of the environment in which they are embedded. In [19], authors address the path planning of multiple robotic searchers to locate a non-adversarial target. A scalable algorithm is proposed to solve the problem efficiently. In [20], authors introduce a weighted visibility graph and propose an optimization framework guaranteeing that each target is tracked by at least one single team member. In [21], authors address the problem of tracking multiple unpredictable targets using a distributed approach. The approach contains both local interaction algorithm and target tracking algorithm. In [22], a switching strategy is proposed for tracking a single target using mobile sensing agents that take bearings-only measurements. In [23], authors model the cooperative localization and target tracking problem within multiple agents as a least squares minimization problem. Their work shows that the problem can be efficiently solved by sparse optimization methods. In [24], authors suggest learning models for target tracking, and further propose a mechanism which reduces the required communications among agents. However, the aforementioned

works do not consider the presence of obstacles in the environment. Since obstacles occlude vision sensors, consideration of them are necessary and important to our problem formulation.

The main contribution of the paper is to present a scalable algorithm for multiple observers and targets that builds on the results of guaranteed tracking strategies provided in [7] for single observer-single target tracking problem in simple environments which models the tracking task as a pursuit-evasion game [25]. An empirical evidence of the efficacy of our algorithm is presented based on simulation results. Finally, experimental results show the feasibility and effectiveness of the proposed strategies.

This paper is outlined as follows. In Sect. 2, we present the problem formulation. In Sect. 3, previous work about a cell decomposition of the workspace based on strategies of the players is described, and the notion of pursuit fields is introduced. In Sect. 4, we propose the moving strategies for pursuers within general polygonal environment and a ranking and aggregate algorithm for allocating pursuers. In Sect. 5, simulation and implementation results are provided to validate the proposed strategies and algorithm. In Sect. 6, we present the conclusions.

2 Problem Description

Consider a planar environment containing polygonal obstacles and two teams of mobile agents. One team of mobile agents, called the *observers* (*pursuers*), are equipped with a vision sensor for surveillance. In this work, we assume that the vision sensor mounted on top of each observer is an omni-directional camera having an infinite range. Thus, the line-of-sight between the target and the observer is assumed to be only obstructed by the presence of obstacles in the environment. The objective of the team of observers is to track another team of mobile agents, called the *targets* (*evaders*). We define the tracking as that each member in the target team should be visible to at least one member in the observer team. Let N_p and N_e denote the numbers of pursuers and evaders, respectively. Throughout this paper, the term "observer" will be used interchangeably with the term "pursuer", and the term "target" will be used interchangeably with the term "evader".

Next, we describe the motion model for the mobile agents. In this work, we assume that all the agents have the following kinematic model

$$\dot{x} = u\cos\theta, \quad \dot{y} = u\sin\theta, \tag{1}$$

where (x, y) denotes the coordinate of the agent in the plane, and u denotes the speed of agent. In this work, we assume that all agents in each team possess the same maximum speed. So let v_p and v_e denote the maximum speed of the pursuer and evader, respectively. Let a denote the ratio of their maximum speeds, i.e., $a = v_e/v_p$. Next, we describe the common information available to agents in each team. All agents in both teams are assumed to have a complete map of the environment,

and are also assumed to know the exact positions of their team mates at all times, including themselves. Finally, we assume that initially each evader is visible to at least one pursuer. Given the initial positions of all the agents, we address the following question: What should be the strategy for the team of pursuers to track all the evaders for the maximum possible time?

In order to consider the worst-case scenarios, we assume that the evaders are adversarial in nature, and try to minimize the time required to escape from the region visible to the team of pursuers. For the simplest case, when all the agents have a complete information about the positions of others, the assumption of an antagonistic team of evaders leads to a two-person zero-sum differential game in between the two teams. In [26], authors have presented a thorough investigation of the problem for the special case when $N_p = N_e = 1$. A complete solution is provided to the problem for specific environments such as the environment containing a semi-infinite obstacle having one corner. However, the complete solution to the aforementioned case in general environments containing polygonal obstacles is still unknown [6]. In this work, we extend previous work to propose algorithms for general environment containing polygonal obstacles that can handle arbitrary number of observers and targets. In the next section, we summarize the main results from [7].

3 Cell Decomposition Around a Corner

In [7], the authors pose the target-tracking problem as a pursuit-evasion game of kind [27] in which the observer is modeled as a pursuer and the evader is modeled as a target. The pursuer intends to maintain the line-of-sight to the target, and the evader would like to break the line-of-sight in finite amount of time. The authors present a cell decomposition of the workspace around a corner based on the strategy used by the winner to ensure a successful outcome. Given the initial position of the pursuer, Fig. 1 depicts the geometry of the individual cells or partitions, and Table 1 provides the strategies for the winners. Figure 2 depicts the same partitions computed

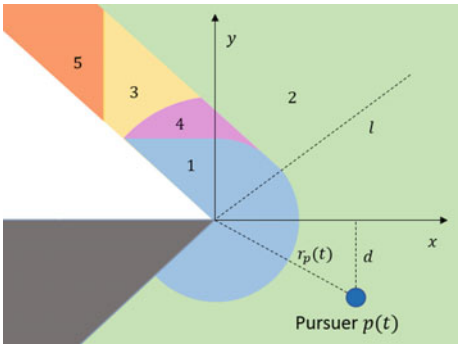


Fig. 1 The environment partition with given pursuer position. The pursuer wins the game in all partitions except for Region 1 (*Blue area*)

Table 1 Policies for the winners

Evader policies	Evader region	Control law
A	1 and $\phi_e \in [\alpha - \pi, \frac{\pi}{2}]$	$\dot{r}_e(t) = -\bar{v}_e$
	1 and $\phi_e \in [\frac{\pi}{2}, \pi + \phi_p]$	$\dot{y}_e(t) = -\bar{v}_e$
Pursuer policies	Evader region	Control law
B	2, 4	$\dot{y}_p(t) = \bar{v}_p$
C	3	$\mathbf{u}_t(t) = \frac{r_p(t)}{r_e(t)} \lvert\mathbf{v}_t(t)\rvert$
		$\mathbf{u}_r(t) = -\frac{r_p(t)}{r_e(t)} \lvert\mathbf{v}_r(t)\rvert$
D	5	$\mathbf{u}_t(t) = \bar{v}_p$

The evader wins the game when it is in region 1, otherwise the pursuer has winning strategies and wins the game

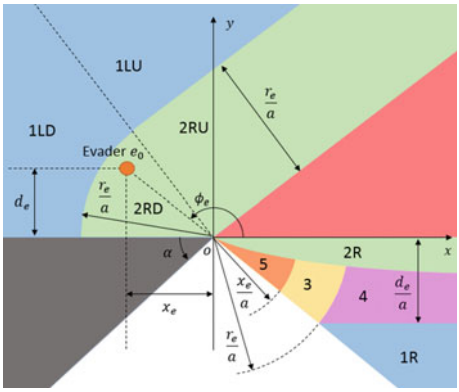


Fig. 2 The partitions computed by fixing evader position

by fixing the position of the evader. The pursuer wins the game in all partitions except for Region 1. However, the strategy used by the pursuer in Region 1 maximizes the time for which it can keep the evader in sight irrespective of the evader's strategy.

Based on the current position of the evader around the corner, the pursuer has an optimal direction of motion that maximizes the time for which the evader is visible depending on the partition in which it is placed. Since the optimal direction of motion at a point can be denoted by a vector, we can generate a vector field in the environment that defines the optimal direction of motion for the pursuer at any time instant. Since the pursuer and the evader are mobile agents, the vector fields are time varying in nature. We use the term *Pursuit Fields* to represent these vector fields. The pursuer can navigate this time varying vector field to optimally track the evader. Figure 3 shows the vector fields for a given position of the evader. Table 2 provides the vector fields generated by the evader in the partitions.

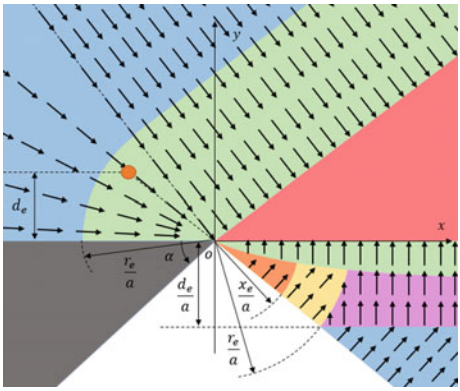


Fig. 3 Pursuit fields with given evader position

Table 2 Vector Fields for optimal navigation around a corner

Region	Vector fields
1LU, 2RU	$\cos\alpha\frac{\partial}{\partial x}-\sin\alpha\frac{\partial}{\partial y}$
1LD, 2RD	$\cos\theta\frac{\partial}{\partial x}-\sin\theta\frac{\partial}{\partial y}$
1R	$-\sin\theta\frac{\partial}{\partial x}+\cos\theta\frac{\partial}{\partial y}$
2R	$\frac{\partial}{\partial y}$
3	$-\sin\theta\frac{\partial}{\partial x}+\cos\theta\frac{\partial}{\partial y}$
4	$\frac{\partial}{\partial y}$
5	$\frac{1}{\sqrt{x^2+y^2}}(-\sin\theta\frac{\partial}{\partial x}+\cos\theta\frac{\partial}{\partial y})$

$\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ represent the basis vectors of the tangent space at a given point

4 Extension to General Environments

4.1 Case 1: $N_p = N_e = 1$

In this subsection, we use the results for a single corner in order to provide navigation strategies for the pursuer in general polygonal environments. From the previous section, one can construct pursuit fields based on the position of the evader around a corner. One can use the pursuit fields to generate pursuit strategies for environments containing multiple obstacles. A plausible way to do so is described next. The presence of an evader in the environment generates a set of pursuit fields, each corresponding to a corner visible to the pursuer around which the evader can escape. In order to generate the guidance law for the pursuer, one can use different metrics to obtain a resultant vector field from the set of pursuit fields. In this work, we use a weighted summation of the individual pursuit fields in order to compute the resultant vector field for pursuer navigation, i.e.,

$$v = \sum_{|C_i|} w_i v_i, \tag{2}$$

where C_i is the ith corner in the environment visible to the pursuer. The vector $\mathbf{w} = [w_1, \ldots, w_k]$ is called the *Risk Vector*. The ith element of the risk vector models the relative risk of the evader breaking the line-of-sight with the pursuer around the ith corner. Define d_i to be the distance between evader and the ith corner, we consider the following metrics for measuring risk:

1. Equidistributed risk: $w_i = \frac{1}{k}$
2. Majority risk: $w_i = 1$, where $i = \arg\max \frac{1}{d_i}$
3. Proportional risk: $w_i = \dfrac{d_i^{-1}}{\sum_{|C_j|} d_j^{-1}}$

The equidistributed risk vector provides equal weight to all the corners visible to the pursuer around which the evader can escape. The majority risk vector only takes into account the pursuit field generated by the corner that is nearest to the evader. The proportional risk vector assigns a weight to each corner that depends on the proximity of the evader from the corner. Figure 4 shows the vector fields that are generated for an environment using the three different techniques. The strategy of the pursuer at any given instant is to navigate along the vector field based on a chosen risk vector. Figure 5 shows the trajectories of the pursuer generated by the three risk

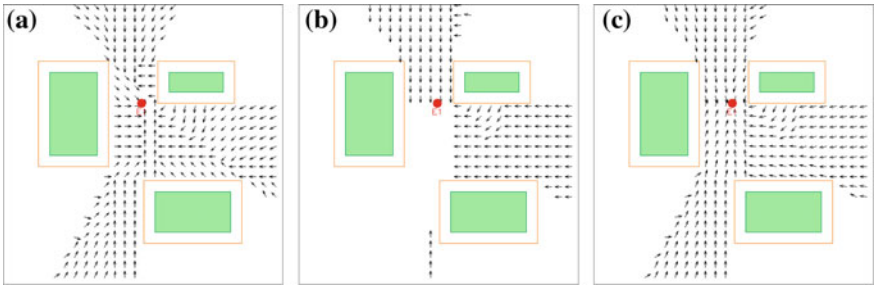


Fig. 4 Vector fields. **a** Equidistributed risk. **b** Majority risk. **c** Proportional risk

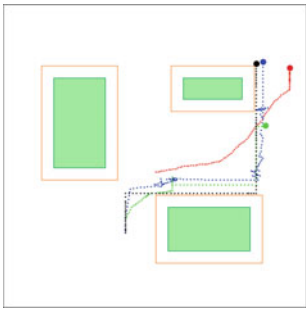


Fig. 5 *Red dots* stand for the trajectory of evader. *Green*, *black* and *blue dots* show the trajectories of pursuers using equidistributed, majority and proportional risk, respectively. Three pursuers are placed at the same place initially

vectors for a fixed evader trajectory. In this paper, we do not compare the three risk vectors because they reflect different situations and their performances largely depend on the environment. Performing comparisons with different risk vectors will be one direction of our future research.

4.2 Case 2: $N_p > 1$, $N_e > 1$

In this section, we present an extension of the previous technique for the case $N_p > 1$, $N_e > 1$. Initially, we discuss the case when all the pursuers use the same risk function to compute the risk associated with a team of targets, following which, we discuss the case when each pursuer may have its own risk function.

First, let us analyze the case when all pursuers have the same risk function. For a given pursuer and evader, one can compute the risk of evasion directly. For two teams of pursuers and evaders, one can compute the risks associated with each pair of pursuer and evader, and use the Hungarian algorithm [28] in order to compute the minimum matching. When pursuer does not see the evader, the risk will be infinity. This works perfectly if $N_p = N_e$. If $N_p > N_e$, we have some pursuers that remain unassigned for the tracking task. In this case, there can be several ways to solve the reassignment problem. In our implementation, for each pursuer that remains unassigned, we let it stay stationary until it is included in the output of Hungarian algorithm. On the other hand, if $N_p < N_e$, there are some evaders that might not be assigned to any pursuer. As long as they are visible to at least one pursuer, they are being tracked by definition.

Next we discuss the case when all pursuers do not use the same risk function. In this case, it is not possible to use the above technique since the values of risk generated by all pursuers may not be comparable. In order to resolve this issue, we propose a ranking and aggregation algorithm. Each pursuer ranks the evaders that are in its field of view. Based on the risk function, pursuer i will have a sequence of risks associated with each evader in its field of view. By sorting this sequence of risks, the pursuer obtains a ranking for the evaders. Therefore, each pursuer will have a ranking of the evaders in its visibility polygon. In order to aggregate the ranking obtained from different pursuers, we use the Borda Count [29] corresponding to each evader. In each ranking, points are allocated to an evader based on its position in the ranking. Finally, the cumulative score for an evader is obtained by adding the points obtained from all the rankings. All the evaders present in the environment can be ranked based on this score. This provides an aggregation scheme to generate a unique ranking of the evaders from the individual preference of the pursuers.

After the aggregation stage, pursuers are allocated to the evaders based on the final scores obtained from the Borda Counts. First, we define $S \in \mathbb{R}^{N_p \times N_e}$ as follows:

$$S_{ij} = \begin{cases} \text{Borda Score of Evader } j & \text{If evader } j \text{ is visible to pursuer } i \\ \infty & \text{If evader } j \text{ is not visible to pursuer } i \end{cases} \tag{3}$$

We apply Hungarian algorithm to S, and obtain an assignment minimizing the total score. Based on the result, we allocate pursuers to the targets. If $N_p \leq N_e$, we have a matching which leaves no pursuer unassigned. However, if $N_p > N_e$, we assign to each unassigned pursuer an evader that is in its field of view, and has the maximum non-infinity Borda score. This reflects that the remaining pursuers try to track the most risky evaders. The complete algorithm is presented in Algorithm 1.

Algorithm 1 Ranking and aggregation algorithm

call BordaCountMethod(r_{ij}) and return $score \in \mathbb{R}^{N_e}$ and $rank \in \mathbb{R}^{N_e}$
declare $S \in \mathbb{R}^{N_p \times N_e}$
for $i = 1 \rightarrow N_p$ **do**
  for $j = 1 \rightarrow N_e$ **do**
    if evader j is visible to pursuer i **then**
      $S_{ij} = score_j$
    else
      $S_{ij} = \infty$
    end if
  end for
end for
Apply Hungarian algorithm to S and assign pursuers accordingly
if $N_p > N_e$ **then**
  for $i = 1 \rightarrow$ number of remaining pursuers **do**
    for $j = 1 \rightarrow N_e$ **do**
      if evader $rank_j$ is visible to pursuer i, and $S_{i\,rank_j} \neq \infty$ **then**
        Assign pursuer i to evader $rank_j$; break;
      end if
    end for
  end for
end if
function BordaCountMethod(r_{ij})
declare $score \in \mathbb{R}^{N_e}$ and $rank \in \mathbb{R}^{N_e}$
for $i = 1 \rightarrow N_p$ **do**
  Rank the ith row of r_{ij} from high to low
  for $j = 1 \rightarrow N_e$ **do**
    $score_k = score_k + (N_e - j)$ where k is the original column index of r_{ij}
  end for
end for
Rank *score* from high to low and save each element's original index to *rank*
return *score* and *rank*

5 Simulation

In this section, we present the simulation results. Simulations are conducted within a 4000 units $\times$4000 units environment with rectangular obstacles. t_1, t_2 and t_3 denote the time for which the pursuers can track the evaders using the three risk vectors

above. In our simulations, we choose $a = 0.4$ and $v_p = 100$. The following simulations are done for random initial positions of the agents with the constraint that every evader is visible to at least one pursuer.

In the first simulation, we consider using the same risk function for computing the risk of each pair of pursuer and evader. The risk function is described as follows

$$r_{ij} = \sum_{|C_k|} d_{jk}^{-1}. \tag{4}$$

where C_k is the kth corner which is visible to both pursuer i and evader j, d_{jk} denotes the distance between evader j and corner k. Figures 6, 7 and 8 show the average values of t_1, t_2 and t_3 for 1000 simulations as we change the number of pursuers and evaders in the environment. These histograms indicate that the proposed strategies offer pursuers good tracking performances. When $N_p \geq N_e$, all the values of t_1 are greater than 50 time units. The minimum average value of t_1 is 74.62 when $N_p = 1$ and $N_e = 4$, which shows that the strategies can still work when more evaders present. It can be seen that t_2 and t_3 are much longer than t_1, which indicates the performance to some extent depends on the selected risk vector. Furthermore, as the number of

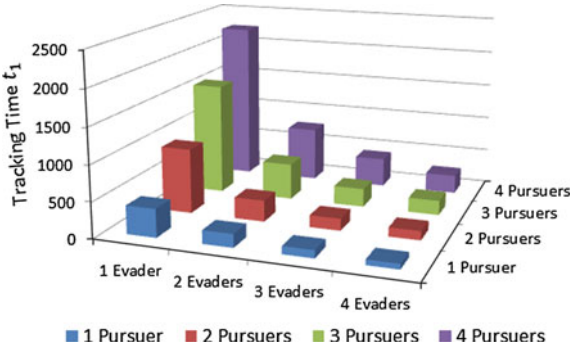


Fig. 6 Tracking time t_1 of pursuers using equidistributed risk vector

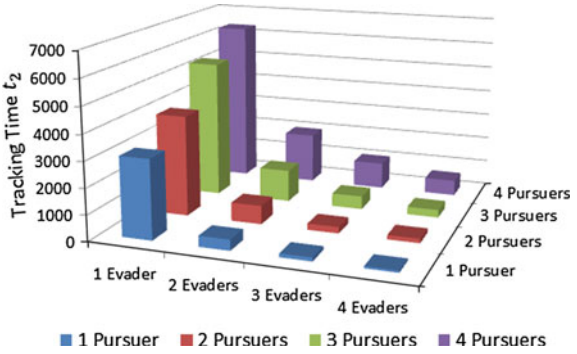


Fig. 7 Tracking time t_2 of pursuers using majority risk vector

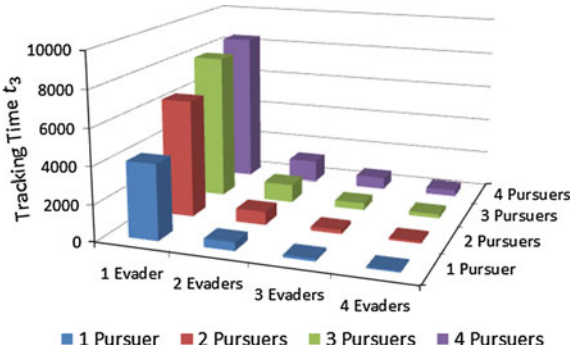


Fig. 8 Tracking time t_3 of pursuers using proportional risk vector

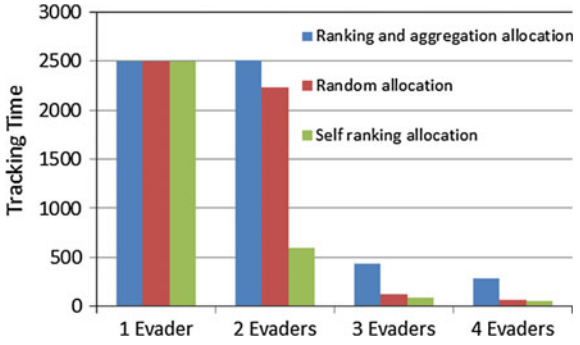


Fig. 9 Tracking time with different allocations

agents increases, the variances between t_1, t_2 and t_3 increase, which means using different risk vectors has a growing influence on the tracking time.

In the second simulation, we consider the case of three pursuers using three disparate risk functions. Figure 9 shows the average tracking time of using ranking and aggregation algorithm for 1000 simulations. As a reference, we arrange another two teams of pursuers in the game. Each team has three pursuers, using the same risk functions with the first team. Pursuers in one team are assigned randomly to their visible targets, pursuers in the other team are assigned to the most risky evader according to their own ranking results. When $N_e = 1$, all the teams of pursuers reach the maximum time we count. But in the cases of multiple evaders, especially when $N_p \geq N_e$, ranking and aggregation algorithm shows a better performance.

As shown in Fig. 10, implementations of above strategies with real robots are done in a 4000 mm $\times$ 4000 mm plane. The results and videos can be found at http://mengzhez.public.iastate.edu/Research/mvbttg.htm. The implementation results are influenced by some practical factors such as robot calibration error and communication delay. But the experimental results mainly match the simulation results which reflect the feasibility of proposed strategy in reality.

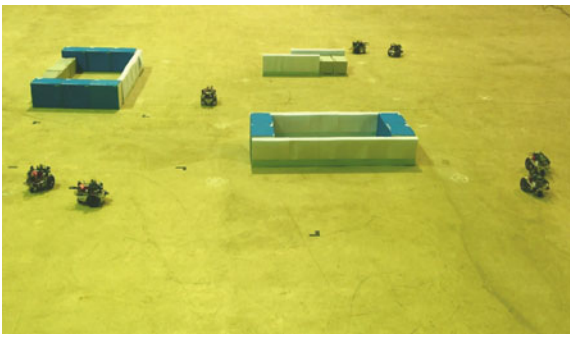


Fig. 10 Implementation set-up

6 Conclusion

In this paper, the problem of visibility-based target tracking for a team of mobile observers trying to track a team of mobile targets was addressed. Initially, we introduced the notion of *pursuit fields* for a single observer to track a single target around a corner based on the results in [7]. We used the pursuit fields to generate navigation strategies for a single observer. In order to tackle the scenario when more than one observer or target is present, we proposed a hierarchical approach. At first a ranking and aggregation technique was used for allocating each observer to a target. Subsequently, each observer computed its navigation strategy based on the results of the single observer-single target problem, thereby, decomposing a large multi-agent problem into numerous 2-agent problems. Based on the aforementioned analysis, we presented a scalable algorithm that can accommodate an arbitrary number of observers and targets. The performance of this algorithm was evaluated based on simulation and implementation. Future work includes considering other risk vectors and making necessary comparisons in some specific environments. Also, other methods will be considered for ranking and aggregation algorithm to improve the tracking system.

References

1. Hollinger, G.A., Djugash, J., Singh, S.: Target tracking without line of sight using range from radio. Auton. Rob. **32**(1), 1–14 (2012)
2. Li, T.-H.S., Chang, S.-J., Tong, W.: Fuzzy target tracking control of autonomous mobile robots by using infrared sensors. IEEE Trans. Fuzzy Syst. **12**, 491–501 (2004)
3. LaValle, S., Gonzalez-Banos, H., Becker, C., Latombe, J.-C.: Motion strategies for maintaining visibility of a moving target. In: IEEE International Conference on Robotics and Automation, Proceedings, vol. 1, pp. 731–736, Apr 1997
4. Kolling, A., Carpin, S.: Cooperative observation of multiple moving targets: an algorithm and its formalization. Int. J. Robot. Res. **26**(9), 935–953 (2007)
5. Lee, D., Kim, G., Kim, D., Myung, H., Choi, H.-T.: Vision-based object detection and tracking for autonomous navigation of underwater robots. Ocean Eng. **48**, 59–68 (2012)

6. Bhattacharya, S., Hutchinson, S.: Approximation schemes for two-player pursuit evasion games with visibility constraints. In: Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland, June 2008
7. Bhattacharya, S., Hutchinson, S.: A cell decomposition approach to visibility-based pursuit evasion among obstacles. Int. J. Robot. Res. **30**(14), 1709–1727 (2011)
8. Gonzalez-Banos, H., Lee, C.-Y., Latombe, J.-C.: Real-time combinatorial tracking of a target moving unpredictably among obstacles. In: IEEE International Conference on Robotics and Automation. Proceedings. ICRA '02, vol. 2, pp. 1683–1690 (2002)
9. Bandyopadhyay, T., Li, Y., Ang, Jr., M.H., Hsu, D.: Stealth Tracking of an Unpredictable Target Among Obstacles (2004)
10. Bandyopadhyay, T., Li, Y., Ang, Jr., M.H., Hsu, D.: A greedy strategy for tracking a locally predictable target among obstacles. In: IEEE International Conference on Robotics and Automation, ICRA 2006, Proceedings, pp. 2342–2347, May 2006
11. Bandyopadhyay, T., Hsu, D., Ang, J., Marcelo, H.: Motion strategies for people tracking in cluttered and dynamic environments. In: Khatib, O., Kumar, V., Pappas, G. (eds.) Experimental Robotics, Springer Tracts in Advanced Robotics, vol. 54, pp. 463–472. Springer, Berlin Heidelberg (2009)
12. Al-Bluwi, I., Elnagar, A.: Maintaining visibility of a moving target: maximizing escape time versus exposure time. In: 11th International Conference on Control Automation Robotics Vision (ICARCV), pp. 982–987, Dec 2010
13. Anderson, R., Milutinovic, D.: A stochastic approach to dubins feedback control for target tracking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3917–3922, Sept 2011
14. Parker, L.: Distributed algorithms for multi-robot observation of multiple moving targets. Auton. Rob. **12**(3), 231–255 (2002)
15. Frew, E.W., Elston, J.: Target assignment for integrated search and tracking by active robot networks. In: Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, pp. 2354–9, May 2008
16. Jung, B., Sukhatme, G.: Tracking targets using multiple robots: the effect of environment occlusion. Auton. Rob. **13**(3), 191–205 (2002)
17. Jung, B., Sukhatme, G.: Cooperative multi-robot target tracking. In: Gini, M., Voyles, R. (eds.) Distributed Autonomous Robotic Systems, vol. 7, pp. 81–90. Springer, Japan (2006)
18. Jung, B., Sukhatme, G.: Real-time motion tracking from a mobile robot. Int. J. Soc. Robot. **2**(1), 63–78 (2010)
19. Hollinger, G., Singh, S., Djugash, J., Kehagias, A.: Efficient multi-robot search for a moving target. Int. J. Robot. Res. **28**(2), 201–219 (2009)
20. Derenick, J., Spletzer, J., Hsieh, A.: An optimal approach to collaborative target tracking with performance guarantees. J. Intell. Rob. Syst. **56**(1–2), 47–67 (2009)
21. Lee, G., Chong, N., Christensen, H.: Tracking multiple moving targets with swarms of mobile robots. Intell. Serv. Robot. **3**(2), 61–72 (2010)
22. Wu, W., Zhang, F.: A switching strategy for target tracking by mobile sensing agents. J. Commun. **8**(1), 47–54 (2013)
23. Ahmad, A., Tipaldi, G., Lima, P., Burgard, W.: Cooperative robot localization and target tracking based on least squares minimization. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5696–5701, May 2013
24. Xu, Z., Fitch, R., Sukkarieh, S.: Decentralised coordination of mobile robots for target tracking with learnt utility models. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2014–2020, May 2013
25. Chung, T., Hollinger, G., Isler, V.: Search and pursuit-evasion in mobile robotics. Auton. Rob. **31**(4), 299–316 (2011)

26. Bhattacharya, S., Candido, S., Hutchinson, S.: Motion strategies for surveillance. In: Proceedings of Robotics: Science and Systems, Atlanta, GA, USA, June 2007
27. Isaacs, R.: Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization. Dover Publications, Mineola (1965)
28. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Logistics Quart. **2**, 83–97 (1955)
29. Emerson, P.: The original borda count and partial voting. Soc. Choice Welf. **40**(2), 353–358 (2013)

Glider CT: Analysis and Experimental Validation

Dongsik Chang, Wencen Wu and Fumin Zhang

Abstract Underwater gliders are robust ocean sensor platforms characterized by high reliability and endurance. Because of their relatively low speed, the motion of underwater gliders is strongly affected by the ocean current, providing data to estimate the depth averaged flow velocity. The glider computerized tomography (Glider CT) algorithm reconstructs a depth-averaged flow field from the navigation errors accumulated along the glider trajectories. This paper justifies the convergence of the Glider CT algorithm as a row action method solving nonlinear equations previously used for bent-ray ultrasonic CT. The paper also validates the algorithm through experiments where the horizontal motion of underwater gliders under flow is imitated by mobile robots in an indoor lab setting. Both theoretical analysis and experimental results suggest Glider CT as a promising method for marine operations.

Keywords Flow field reconstruction · Underwater gliders

1 Introduction

The underwater glider has found a broad range of applications such as oil field surveys, military operations, and deep-sea and coastal research [10, 13]. The sampling and monitoring performance of gliders significantly relies on the navigation performance of gliders. Because of their relatively low speed, the motion of gliders is sensitive to the ocean current. Therefore, control systems [3, 4, 11] and algorithms [5, 17], and path planning algorithms [3, 4, 15] have been developed to navigate gliders through ocean flow with improved performance.

D. Chang (✉) · F. Zhang
Georgia Institute of Technology, Atlanta, USA
e-mail: dsfrancis3@gatech.edu

F. Zhang
e-mail: fumin@gatech.edu

W. Wu
Rensselaer Polytechnic Institute, Troy, USA
e-mail: wuw8@rpi.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_20

The primary means of localization for underwater gliders is the global positioning system (GPS) [7]. However, since GPS signals cannot propagate through sea water, gliders estimate their underwater positions via dead-reckoning between regular surfacing events for GPS updates. Because gliders swim at relatively low speed, their trajectories are strongly perturbed by the ocean current. Hence, we can typically observe a difference between the dead-reckoning and actual surfacing positions of a glider. We refer to this difference as a *dead-reckoning error*. To mitigate the dead-reckoning error, a glider computes an estimate of average flow velocity along the trajectory between the last and current surfacing positions, and incorporates the flow estimate into navigation until the following surfacing event [8]. However, the estimate does not account for the temporal/spatial variations of the flow field during navigation, and our recent work [3] emphasized the importance of incorporating such variations into navigation algorithms in field deployments.

In [2, 3], we proposed efficient methods for real-time ocean current modeling based on estimates of flow from underwater gliders to improve navigation performance. The method in [3] first approximates slowly-varying non-tidal flow from glider-derived flow estimates and then adds rapidly-varying tidal flow from an tidal ocean model to the non-tidal flow. However, since the non-tidal flow in the method is empirically estimated from only glider-derived flow estimates, its accuracy is limited to a local area around each glider. The method in [2] approximates ocean currents using spatial and temporal basis functions. The ocean model constructed there requires historic data, such as HF-radar observation data or general circulation model output data, to initialize. Once initialized, the model is updated based on the flow estimates from a group of gliders in real-time and provides ocean current data at higher resolution than existing approaches.

In our previous work in [16], we developed the glider computerized tomography (Glider CT) algorithm that reconstructs the spatial distribution of a depth-averaged flow field with no a priori knowledge of the field. Glider CT is named after CT, which reconstructs an image of the internal structure of an object from signals (e.g., X-rays) that are projected onto the object. A typical setup of CT has the transmitters and receivers of signals around an object. The transmitters emit signals onto an object, and while penetrating the object, the signals attenuate. Then, the remaining strength of the signals is measured at the receivers. Based on the signal paths and the measured signal strengths attenuated along the paths, an image of the object is reconstructed. In a similar way to CT, Glider CT reconstructs a flow field from the trajectories and dead-reckoning errors of gliders. We draw analogies between signal paths of CT and glider trajectories of Glider CT and between measured signal strengths attenuated along the signal paths and dead-reckoning errors accumulated along the glider trajectories.

The structure of the Glider CT algorithm is very similar to a general CT reconstruction algorithm for bent ultrasound-rays [14]. In this paper, we extend our work in [16] by analyzing the Glider CT algorithm and providing a convergence proof, which is applicable to the method in [14] as well. Our algorithm solves a specific type of nonlinear systems of equations by extending the Kaczmarz method for linear equations. The Kaczmarz method is one of the row-action methods [1]. Convergence

results have previously been obtained [6, 9] regarding various Kaczmarz-type methods for nonlinear equations. The glider CT algorithm can be viewed as one special case of these methods, and our proof of convergence adds to the collection.

In addition to the convergence analysis, this paper provides experimental results demonstrating the Glider CT algorithm. We imitate the horizontal motion of underwater gliders under a flow field using Khepera III robots under a simulated flow field. We place a light source in a target domain and simulate a flow field based on light intensity. By applying the dead-reckoning technique of gliders, we estimate the dead-reckoning trajectories of Khepera III robots. We control the motion of Khepera III robots as if their trajectories are affected by the simulated flow field. Since the actual underwater trajectories of gliders are unknown because of unavailable GPS signals underwater, we treat the trajectories of Khepera III robots as unknown. Only the starting and ending positions of the robots are used to compute the dead-reckoning error. Then, we apply the Glider CT algorithm to reconstruct the simulated flow field. The experimental results show a promising performance of the algorithm in practical settings.

The rest of the paper is organized as follows. Section 2 provides background information about underwater glider navigation, and Sect. 3 reviews our preliminary work on the Glider CT algorithm. In Sects. 4 and 5, we analyze the details of Glider CT and prove the convergence of the algorithm, respectively. Section 6 validates the algorithm through experiments, and Sect. 7 concludes the paper.

2 Background: Underwater Glider Navigation

An underwater glider regularly comes to the surface of water for GPS updates and data transfer. Between two surfacing events, glider navigation consists of two phases: surface and subsurface. We denote the actual and dead-reckoning positions of a glider at time t by $\mathbf{r}(t)$ and $\tilde{\mathbf{r}}(t)$, respectively. We also denote the time associated with the kth surfacing and diving events by t_k^s and t_k^d, respectively. Figure 1 illustrates glider navigation from the $(k-1)$th diving event to the kth diving event.

In this paper, we deal with the glider and the flow in the horizontal plane. Suppose we navigate a glider towards a waypoint during the kth subsurface phase. Before it dives, the glider computes its heading θ_k towards the waypoint. Then, the glider dives at $\mathbf{r}(t_{k-1}^d)$ and navigates underwater until it reaches the waypoint by dead-reckoning, which estimates the position $\tilde{\mathbf{r}}(t)$ of the glider using estimates of glider speed, compass heading, and flow velocity. Because of the influence of flow, when the glider comes back to the surface of water at the kth surfacing event, the glider experiences the dead-reckoning error, which is the difference between the dead-reckoning surfacing position $\tilde{\mathbf{r}}(t_k^s)$ and the GPS surfacing position $\mathbf{r}(t_k^s)$.

Upon the kth surfacing event, a glider computes an estimate of average flow velocity along the glider trajectory based on the dead-reckoning error accumulated over the $(k-1)$th subsurface phase. This glider-derived flow estimate can be either incorporated into navigation to reduce the dead-reckoning error or deactivated so

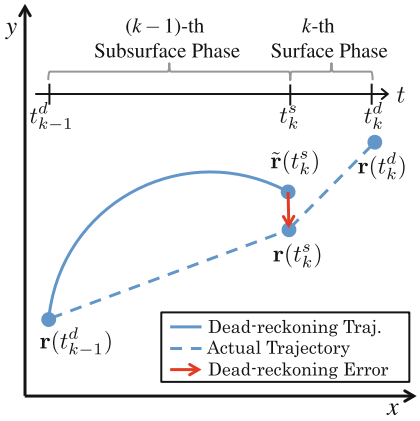


Fig. 1 Glider navigation during the $(k-1)$th subsurface phase followed by the kth surface phase. The figure shows the actual (*blue dashed line*) and dead-reckoning (*blue solid line*) trajectories of a glider. The dead-reckoning error is shown as a *red arrow*

that no flow estimate is used in navigation. Let us introduce a switching signal I_f to indicate whether the estimated flow is used for navigation or not. The signal $I_\mathsf{f} = 1$ indicates that the estimated flow is used for navigation and 0, otherwise. Then, the glider-derived flow estimate at the kth surfacing event is given by

$$\bar{\mathbf{f}}_k = \bar{\mathbf{f}}_{k-1} I_\mathsf{f} + \frac{\mathbf{r}(t^s_k) - \tilde{\mathbf{r}}(t^s_k)}{t^s_k - t^d_{k-1}}, \tag{1}$$

which combines the previous flow estimate used to navigate over the $(k-1)$th subsurface phase with the new flow estimate based on the dead-reckoning error accumulated during the $(k-1)$th subsurface phase.

To describe the motion of a glider in the plane, we use a particle model with a constant through-water speed s_h. The position of a glider along the dead-reckoning trajectory between the kth and $(k+1)$th surfacing events can be predicted by integrating the following equation over time:

$$\dot{\tilde{\mathbf{r}}}(t) = s_\mathsf{h} \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} + \bar{\mathbf{f}}_k I_\mathsf{f}. \tag{2}$$

However, the real flow experienced by a glider may be different from the glider-estimated flow. Hence, the actual trajectory can be described by integrating the following equation over time:

$$\dot{\mathbf{r}}(t) = s_\mathsf{h} \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} + \mathbf{f}(\mathbf{r}, t) = \dot{\tilde{\mathbf{r}}}(t) + \mathbf{f}(\mathbf{r}, t) - \bar{\mathbf{f}}_k I_\mathsf{f}, \tag{3}$$

which is usually unknown because of the unknown flow velocity $\mathbf{f}(\mathbf{r}, t)$.

3 Preliminary Work: Problem Formulation of Glider CT

The Glider CT problem is formulated from the fact that the dead-reckoning error accumulated along the glider trajectory is determined by a line integral of the difference between real flow experienced by the glider and glider-estimated flow incorporated in navigation. Suppose we deploy m gliders in the ocean and consider glider navigation over one subsurface phase only. Hereafter, we drop the subscript index k for the surfacing events and use the subscript index $i = \{1, \ldots, m\}$ for the gliders for simplicity. After each glider finishes one subsurface phase, we will obtain a dead-reckoning error $\mathbf{d}_i$ and a glider-derived flow estimate $\bar{\mathbf{f}}_i$ from each glider. The dead-reckoning error accumulates over one subsurface phase, and from Eqs. (2) and (3), it is given by

$$\mathbf{d}_i = \int_{t_i^d}^{t_i^s} \left(\dot{\mathbf{r}}_i(\tau) - \dot{\tilde{\mathbf{r}}}_i(\tau) \right) d\tau = \int_{t_i^d}^{t_i^s} \mathbf{f}(\mathbf{r}_i) d\tau - \bar{\mathbf{f}} \cdot I_{\mathsf{f}} \left(t_i^s - t_i^d \right), \tag{4}$$

in which t_i^d and t_i^s are the diving and surfacing times of the ith glider. We introduce arc-length parameter l of the trajectory, given by

$$dl = s_{\mathsf{tr}} dt, \tag{5}$$

in which s_{tr} is the speed of the glider along its actual trajectory, which satisfies

$$s_{\mathsf{tr}}(\mathbf{r}_i) = \|\dot{\mathbf{r}}_i\| = \left\| s_{\mathsf{h}} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} + \mathbf{f}(\mathbf{r}_i) \right\|. \tag{6}$$

Substituting Eqs. (5) and (6) into Eq. (4), we derive

$$\mathbf{d}_i = \int_C \frac{1}{s_{\mathsf{tr}}(\mathbf{r}_i)} \mathbf{f}(\mathbf{r}_i) dl - \bar{\mathbf{f}} \cdot I_{\mathsf{f}} \left(t_i^s - t_i^d \right). \tag{7}$$

Since the second term on the right side of the equation is known, for simplicity we let $I_{\mathsf{f}} = 0$ throughout the paper. However, our results apply to the general case without requiring $I_{\mathsf{f}} = 0$.

Suppose we navigate gliders in area A. Let us discretize A into $R \times S$ grids with $A_{(r,s)}$ referring to the (r, s)th grid. We denote the flow velocity in each cell by $\mathbf{f}_j$, $j = \{1, \ldots, n = RS\}$. Indices j, r, and s satisfy $j = (r-1)S + s$. For the ith glider passing through the jth grid, we denote the length of the trajectory in the cell by $L_{(i,j)}$. Since we have a constant flow in each grid, the glider speed along the trajectory is given by

$$s_{\mathsf{tr}}^{(i,j)}(\mathbf{f}_j) = \left\| s_{\mathsf{h}} \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} + \mathbf{f}_j \right\|.$$

We assume that the horizontal projection of the glider trajectory is straight and the heading θ_i is constant along the trajectory. Then, the discretized version of Eq. (7) with $I_{\mathsf{f}} = 0$ is

$$\mathbf{d}_i = \sum_{j=1}^{n} \frac{L_{(i,j)}}{s_{\mathsf{tr}}^{(i,j)}(\mathbf{f}_j)} \mathbf{f}_j, \quad i = \{1, \ldots, m\}.$$

Now, consider the flow velocity along the x and y directions separately. For $i = \{1, \ldots, m\}$, we can write

$$d_{x,i} = \sum_{j=1}^{n} \frac{L_{(i,j)}}{s_{\mathsf{tr}}^{(i,j)}(\mathbf{f}_j)} f_{x,j}, \qquad d_{y,i} = \sum_{j=1}^{n} \frac{L_{(i,j)}}{s_{\mathsf{tr}}^{(i,j)}(\mathbf{f}_j)} f_{y,j}. \tag{8}$$

By introducing vectors $\mathbf{d}_x = [d_{x,1}, d_{x,2}, \ldots, d_{x,m}]^T$, $\mathbf{d}_y = [d_{y,1}, d_{y,2}, \ldots, d_{y,m}]^T$, $\mathbf{f}_x = [f_{x,1}, f_{x,2}, \ldots, f_{x,n}]^T$, and $\mathbf{f}_y = [f_{y,1}, f_{y,2}, \ldots, f_{y,n}]^T$, we can rewrite Eq. (8) as

$$\mathbf{d}_x = \mathbf{L}(\mathbf{f})\mathbf{f}_x, \qquad \mathbf{d}_y = \mathbf{L}(\mathbf{f})\mathbf{f}_y, \tag{9}$$

where

$$\mathbf{L}(\mathbf{f}) = \begin{bmatrix} \frac{L_{(1,1)}}{s_{\mathsf{tr}}^{(1,1)}(\mathbf{f}_1)} & \cdots & \frac{L_{(1,n)}}{s_{\mathsf{tr}}^{(1,n)}(\mathbf{f}_n)} \\ \vdots & \ddots & \vdots \\ \frac{L_{(m,1)}}{s_{\mathsf{tr}}^{(m,1)}(\mathbf{f}_1)} & \cdots & \frac{L_{(m,n)}}{s_{\mathsf{tr}}^{(m,n)}(\mathbf{f}_n)} \end{bmatrix}, \tag{10}$$

which is nonlinear and typically underdetermined ($m < n$). By solving Eq. (9) for $\mathbf{f}_x$ and $\mathbf{f}_y$, we can estimate flow.

4 Analysis of Glider CT

To solve the underdetermined and nonlinear system of equations in Eq. (9), we developed the Glider CT algorithm (Algorithm 1) that iteratively updates a solution to the equations with relaxation parameter $\lambda^{(k,i-1)}$. The relaxation parameter affects the convergence rate of the algorithm, and for simplicity, we assume $\lambda^{(k,i-1)} = 1$, $\forall k, i$ here. Let us omit x and y in the system for now. Given a nonlinear system

$$\mathbf{L}(\mathbf{f})\mathbf{f} = \mathbf{d}, \tag{11}$$

where

$$\mathbf{L}(\mathbf{f}) = \begin{bmatrix} \mathbf{L}_1(\mathbf{f}) \\ \vdots \\ \mathbf{L}_m(\mathbf{f}) \end{bmatrix}, \ \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \ \mathbf{d} = \begin{bmatrix} d_1 \\ \vdots \\ d_m \end{bmatrix},$$

the Glider CT algorithm finds a solution to the system equations in an iterative way. For the kth iteration, let $\mathbf{f}^{(k,0)} = \left(f_1^{(k,0)}, f_2^{(k,0)}, \ldots, f_n^{(k,0)}\right)^T$ be the initial solution, and let us divide a new solution into m sequences given by

$$\begin{aligned}\mathbf{f}^{(k,1)} &= \left(f_1^{(k,1)}, f_2^{(k,1)}, \ldots, f_n^{(k,1)}\right)^T, \\ &\vdots \\ \mathbf{f}^{(k,m)} &= \left(f_1^{(k,m)}, f_2^{(k,m)}, \ldots, f_n^{(k,m)}\right)^T.\end{aligned}$$

Since the system (11) is underdetermined, there may exist infinitely many solutions. Suppose given $\mathbf{f}^{(k,i-1)}$, we want to update solution $\mathbf{f}^{(k,i)}$ to the system in the form of

$$\mathbf{f}^{(k,i)} = \mathbf{f}^{(k,i-1)} + \alpha_i \left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T, \quad i = \{1, \ldots, m\}, \tag{12}$$

which updates the solution by adding $\left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T$ weighted by α_i sequentially from $i = 1$ to m for the kth iteration.

Algorithm 1: Glider CT

Data: Dead-reckoning errors $\mathbf{d}_i, i = \{1, \ldots, m\}$

Set $k = 0$. Make an initial guess of the solution $\mathbf{f}_x^{(k+1,0)}$ and $\mathbf{f}_y^{(k+1,0)}$.

repeat

  Let $k = k + 1$.

  for $i = 1$ **to** m **do**

    Update the solution by

$$\mathbf{f}_x^{(k,i)} = \mathbf{f}_x^{(k,i-1)} + \lambda^{(k,i-1)} \frac{d_{x,i} - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}_x^{(k,i-1)}}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2} \left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T,$$

$$\mathbf{f}_y^{(k,i)} = \mathbf{f}_y^{(k,i-1)} + \lambda^{(k,i-1)} \frac{d_{y,i} - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}_y^{(k,i-1)}}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2} \left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T.$$

  end

  Let $\mathbf{f}_x^{(k+1,0)} = \mathbf{f}_x^{(k,m)}$ and $\mathbf{f}_y^{(k+1,0)} = \mathbf{f}_y^{(k,m)}$.

until *a stopping condition is met*

Let us define a residual term

$$r^{(k,i-1)}(\mathbf{f}) = \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f} - d_i. \tag{13}$$

To find α_i in Eq. (12), we let $r^{(k,i-1)}(\mathbf{f}) = 0$ at $\mathbf{f} = \mathbf{f}^{(k,i)}$ and substitute Eq. (12) into Eq. (13), which yields

$$\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)} + \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T \alpha_i - d_i = 0.$$

Assuming $\mathbf{L}_i\mathbf{L}_i^T \neq 0$, the equation has the unique solution

$$\alpha_i = \frac{d_i - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)}}{\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T} = \frac{d_i - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)}}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2}. \tag{14}$$

Substituting Eq. (14) into Eq. (12), we have

$$\mathbf{f}^{(k,i)} = \mathbf{f}^{(k,i-1)} + \frac{d_i - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)}}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2}\left(\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T, \tag{15}$$

which is used to update the solution in Algorithm 1 with $\lambda^{(k,i-1)} = 1$. Once we have $\mathbf{f}^{(k,m)}$, we obtain the initial sequence for the $(k+1)$th iteration by $\mathbf{f}^{(k+1,0)} = \mathbf{f}^{(k,m)}$.

5 The Convergence of the Glider CT Algorithm

Given a nonlinear system (11), we claim that the solution to the system equations derived from the Glider CT algorithm $\mathbf{f}^{(k,i)} = \left(f_1^{(k,i)}, f_2^{(k,i)}, \ldots, f_n^{(k,i)}\right)$, $k = \{1, 2, \ldots\}$, $i = \{1, \ldots, m\}$ converges to the true solution $\mathbf{f}^* = \left(f_1^*, f_2^*, \ldots, f_n^*\right)$. Suppose there exists a ball $\mathscr{B}(\mathbf{f}^*, \delta)$ around $\mathbf{f}^*$ with radius $\delta > 0$ where the following two assumptions hold for all $\mathbf{f} \in \mathscr{B}(\mathbf{f}^*, \delta)$:

Assumption 1 $\mathbf{L}_i(\mathbf{f})$ is Lipschitz continuous for all i with the largest Lipschitz constant γ, i.e., given Lipschitz constant $\gamma_i > 0$ for $\mathbf{L}_i(\mathbf{f})$, $i = \{1, \ldots, m\}$, $\gamma = \max_i \gamma_i$.

Assumption 2 There exists $\varepsilon > 0$ that satisfies the following:

(1) $\lambda_{\max}\left(I - \mathbf{L}_i^+(\mathbf{f})\mathbf{L}_i(\mathbf{f})\right) < 1 - \varepsilon$ for all i, where $\lambda_{\max}(\cdot)$ is the largest eigenvalue,

(2) $\dfrac{\gamma\|\mathbf{f}^*\|}{\|\mathbf{L}_i(\mathbf{f})\|} < \sqrt{\varepsilon}$ for all i.

Let us define $\mathbf{L}_i^+(\mathbf{f}) = \frac{\mathbf{L}_i(\mathbf{f})^T}{\|\mathbf{L}_i(\mathbf{f})\|^2}$, referred to as the pseudoinverse of $\mathbf{L}_i(\mathbf{f})$ in this paper.

Lemma 1 *$\mathbf{L}_i^+(\mathbf{f})$ satisfies the following four conditions for the Moore-Penrose pseudoinverse [12]:*

1. $\mathbf{L}_i(\mathbf{f})\mathbf{L}_i^+(\mathbf{f})\mathbf{L}_i(\mathbf{f}) = \mathbf{L}_i(\mathbf{f})$
2. $\mathbf{L}_i^+(\mathbf{f})\mathbf{L}_i(\mathbf{f})\mathbf{L}_i^+(\mathbf{f}) = \mathbf{L}_i^+(\mathbf{f})$
3. $(\mathbf{L}_i(\mathbf{f})\mathbf{L}_i^+(\mathbf{f}))^T = \mathbf{L}_i(\mathbf{f})\mathbf{L}_i^+(\mathbf{f})$
4. $(\mathbf{L}_i^+(\mathbf{f})\mathbf{L}_i(\mathbf{f}))^T = \mathbf{L}_i^+(\mathbf{f})\mathbf{L}_i(\mathbf{f})$

Proof By simply substituting $\mathbf{L}_i^+(\mathbf{f})$ into the above four conditions, we can show that Lemma 1 holds. □

Let us denote the Euclidean distance between $\mathbf{p}$ and $\mathbf{q}$ by $\mathrm{dist}(\mathbf{p}, \mathbf{q})$. In the following theorem, we prove the convergence of the Glider CT algorithm.

Theorem 1 *Suppose there exists a ball* $\mathscr{B}(\mathbf{f}^*, \delta)$ *around* $\mathbf{f}^*$ *with radius* $\delta > 0$ *where Assumptions 1 and 2 hold. Starting from any initial point* $\mathbf{f}^{(1,0)}$ *within the ball, e.g.,* $\mathrm{dist}(\mathbf{f}^*, \mathbf{f}^{(1,0)}) < \delta$, *the sequence* $\mathbf{f}^{(k,i)}$ *generated by Algorithm 1 converges to* $\mathbf{f}^*$.

Proof Let us define an error term $\mathbf{e}^{(k,i)} = \mathbf{f}^{(k,i)} - \mathbf{f}^*$. By subtracting $\mathbf{f}^*$ from the both sides of Eq. (15) and substituting $\mathbf{L}_i^+(\mathbf{f})$ and $r^{(k,i-1)}(\mathbf{f}^{(k,i-1)})$, we have

$$\mathbf{e}^{(k,i)} = \mathbf{e}^{(k,i-1)} - \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) r^{(k,i-1)}(\mathbf{f}^{(k,i-1)}), \;\; i = \{1, \ldots, m\},$$

The square of the Euclidean norm of the error is

$$\begin{aligned}\left\langle \mathbf{e}^{(k,i)}, \mathbf{e}^{(k,i)} \right\rangle = \left\langle \mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)} \right\rangle - 2 \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) r^{(k,i-1)}(\mathbf{f}^{(k,i-1)}), \mathbf{e}^{(k,i-1)} \right\rangle \\ + \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) r^{(k,i-1)}(\mathbf{f}^{(k,i-1)}), \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) r^{(k,i-1)}(\mathbf{f}^{(k,i-1)}) \right\rangle. \quad (16)\end{aligned}$$

Since $d_i = \mathbf{L}_i(\mathbf{f}^*)\mathbf{f}^*$, we can express residual $r^{(k,i-1)}(\mathbf{f}^{(k,i-1)})$ as

$$\begin{aligned} r^{(k,i-1)}(\mathbf{f}^{(k,i-1)}) &= \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)} - d_i \\ &= \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)} - \mathbf{L}_i(\mathbf{f}^*)\mathbf{f}^* \\ &= \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^{(k,i-1)} - \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^* + \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{f}^* - \mathbf{L}_i(\mathbf{f}^*)\mathbf{f}^* \\ &= \mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{e}^{(k,i-1)} + \mathbf{h}^{(k,i-1)}\mathbf{f}^*, \end{aligned} \quad (17)$$

where we define $\mathbf{h}^{(k,i-1)} = \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) - \mathbf{L}_i(\mathbf{f}^*)$. By substituting $r^{(k,i-1)}(\mathbf{f}^{(k,i-1)})$ in Eq. (17) into Eq. (16), we have

$$\begin{aligned}\left\langle \mathbf{e}^{(k,i)}, \mathbf{e}^{(k,i)} \right\rangle = \left\langle \mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)} \right\rangle - 2 \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{h}^{(k,i-1)} \mathbf{f}^*, \mathbf{e}^{(k,i-1)} \right\rangle \\ - 2 \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)} \right\rangle \\ + \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)}, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)} \right\rangle \\ + 2 \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{h}^{(k,i-1)} \mathbf{f}^*, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)} \right\rangle \\ + \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{h}^{(k,i-1)} \mathbf{f}^*, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{h}^{(k,i-1)} \mathbf{f}^* \right\rangle. \end{aligned} \quad (18)$$

By the property of the inner product and Lemma 1, the fourth and fifth terms on the right side of Eq. (18) become

$$\begin{aligned} &\left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)}, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)} \right\rangle \\ &= \left\langle \left(\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \right)^T \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)} \right\rangle \\ &= \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) \mathbf{L}_i(\mathbf{f}^{(k,i-1)}) \mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)} \right\rangle \end{aligned} \quad (19)$$

and

$$\begin{aligned}&\left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\mathbf{e}^{(k,i-1)}\right\rangle \\ &= \left\langle \left(\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)^T \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*, \mathbf{e}^{(k,i-1)}\right\rangle \\ &= \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*, \mathbf{e}^{(k,i-1)}\right\rangle, \end{aligned} \tag{20}$$

respectively. By substituting Eqs. (19) and (20) into Eq. (18), we have

$$\begin{aligned}\left\langle \mathbf{e}^{(k,i)}, \mathbf{e}^{(k,i)}\right\rangle = &\left\langle \left(I - \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)\mathbf{e}^{(k,i-1)}, \mathbf{e}^{(k,i-1)}\right\rangle \\ &+ \left\langle \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*, \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*\right\rangle. \end{aligned} \tag{21}$$

Let us define $\langle x, y\rangle_G = \langle Gx, y\rangle$ as an inner product of $x, y \in \mathbb{R}^n$ induced by matrix G and $\|x\|_G = \sqrt{\langle x, x\rangle_G}$ as a norm of $x \in \mathbb{R}^n$ induced by $\langle \cdot, \cdot\rangle_G$. Then, Eq. (21) can be rewritten as

$$\|\mathbf{e}^{(k,i)}\|^2 = \|\mathbf{e}^{(k,i-1)}\|^2_{\left(I - \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)} + \|\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{h}^{(k,i-1)}\mathbf{f}^*\|^2.$$

Since $\mathbf{L}_i(\mathbf{f})$ is Lipschitz continuous,

$$\|\mathbf{h}^{(k,i-1)}\| = \|\mathbf{L}_i(\mathbf{f}^{(k,i-1)}) - \mathbf{L}_i(\mathbf{f}^*)\| \le \gamma\|\mathbf{f}^{(k,i-1)} - \mathbf{f}^*\| = \gamma\|\mathbf{e}^{(k,i-1)}\|,$$

in which γ is the Lipschitz constant in Assumption 1. Then, we have

$$\begin{aligned}\|\mathbf{e}^{(k,i)}\|^2 \le &\lambda_{\max}\left(I - \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right)\|\mathbf{e}^{(k,i-1)}\|^2 \\ &+ \gamma^2\|\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\|^2\|\mathbf{e}^{(k,i-1)}\|^2\|\mathbf{f}^*\|^2.\end{aligned}$$

Since $\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)}) = \frac{\mathbf{L}_i(\mathbf{f}^{(k,i-1)})^T}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2}$, $\|\mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\|^2 = \frac{1}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2}$, which gives

$$\begin{aligned}\|\mathbf{e}^{(k,i)}\|^2 &\le \left(\lambda_{\max}\left(I - \mathbf{L}_i^+(\mathbf{f}^{(k,i-1)})\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\right) + \frac{\gamma^2\|\mathbf{f}^*\|^2}{\|\mathbf{L}_i(\mathbf{f}^{(k,i-1)})\|^2}\right)\|\mathbf{e}^{(k,i-1)}\|^2 \\ &= \nu^{(k,i-1)}\|\mathbf{e}^{(k,i-1)}\|^2 \; k = \{1, 2, \ldots\}, \; i = \{1, \ldots, m\}, \end{aligned} \tag{22}$$

where $\mathbf{e}^{(k,i)} = \mathbf{f}^* - \mathbf{f}^{(k,i)}$. Since we set the initial solution for each iteration to be the last sequence of the solution from the previous iteration, e.g., $\mathbf{f}^{(k+1,0)} = \mathbf{f}^{(k,m)}$ for the $(k+1)$th iteration, we can express Eq. (22) as $\|\mathbf{e}^s\|^2 \le \nu^{s-1}\|\mathbf{e}^{s-1}\|^2$, where $s = (k-1)m + i$ corresponds to (k, i). Since $\mathbf{f}^{(1,0)} = \mathbf{f}^0 \in \mathscr{B}(\mathbf{f}^*, \delta)$ where Assumptions 1 and 2 hold, $\nu^0 < 1$, and therefore $\mathbf{f}^1 \in \mathscr{B}(\mathbf{f}^*, \delta)$. This applies to the following iterations sequentially, which leads to $\nu^1, \nu^2, \ldots < 1$ and $\mathbf{f}^2, \mathbf{f}^3, \ldots \in \mathscr{B}(\mathbf{f}^*, \delta)$. Therefore, starting from $\mathbf{f}^{(1,0)} \in \mathscr{B}(\mathbf{f}^*, \delta)$, we have $\nu^{(k,i)} < 1$ and $\mathbf{f}^{(k,i)} \in \mathscr{B}(\mathbf{f}^*, \delta)$ for all k and i. Hence, $\mathbf{e}^{(k,i)} \to 0$, and $\mathbf{f}^{(k,i)} \to \mathbf{f}^*$. □

6 Experimental Results

We validate the Glider CT algorithm through experiments using Khepera III robots. The experimental setup is shown in Fig. 2a. The starting and ending positions of robots are identified by a camera installed on top of the experimental domain. Even though the actual trajectory of a robot is observable through the camera, we treat the nominal trajectory of a robot as unknown to us since the underwater trajectory of a glider is unknown in glider operations. To construct a flow field, we place a light source at the left bottom corner $(x, y) = (0, 0)$ of a domain and simulate a 2-D flow field such that all the flow vectors are in the direction of $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$ and their magnitudes are scaled by the light intensity throughout the domain shown in Fig. 2b. The intensity of ambient light around Khepera III robots is measured by 9 IR sensors located on the side of each robot. The measurements of the light intensity range from 0 to $I_{\max} = 4096$, where a lower value indicates higher light intensity.

The horizontal motion of underwater gliders is imitated using Khepera III robots under a simulated flow field. Given an initial heading θ of a robot, its dead-reckoning trajectory is computed by integrating the dead-reckoning motion of gliders in Eq. (2). As discussed in Sect. 3, we set $I_f = 0$. The nominal motion of a Khepera III robot is implemented following Algorithm 2. At step k, each robot first measures the intensity of ambient light from the nine IR sensors at the current positions of the robots and computes the mean of the sensor measurements. To make a lower mean value of the sensor measurements represent a lower light intensity, we subtract the mean of the measurements from I_{max} and compute a ratio of the mean light intensity around each robot. Then, we obtain the magnitude of the simulated flow at the current positions of the robots by scaling the ratio with scaling factor c. The simulated flow field is constructed by multiplying the magnitude by direction vector $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$.

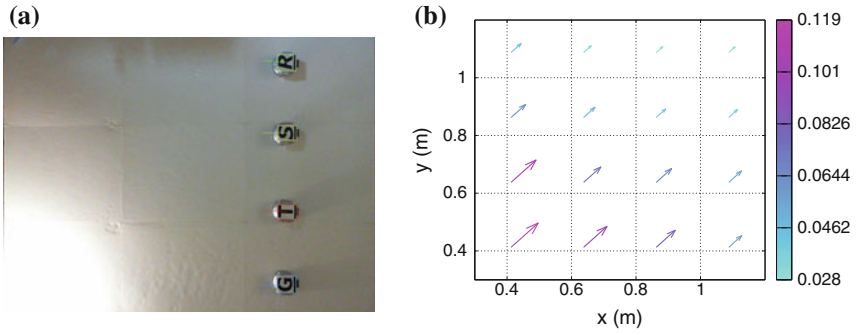


Fig. 2 Experimental setup with Khepera III robots in a light field. **a** Four Khepera III robots are differentiated using letters 'G','T','S', and 'R'. National Instruments LabVIEW identifies the positions (the colored *rectangle* around each robot) and headings (the *line* attached to each robot) of the robots. A light source is located at the *left bottom* corner to simulate a flow field. **b** The simulated true flow field of the domain

We compute the motion $\dot{\mathbf{r}} = [\dot{r}_x, \dot{r}_y]^T$ of a Khepera III robot under the simulated flow using Eq. (3). To implement motion $\dot{\mathbf{r}}$ in a Khepera III robot, we decompose $\dot{\mathbf{r}}$ into its magnitude and angle, i.e., $\dot{\mathbf{r}} = \|\dot{\mathbf{r}}\| \angle \dot{\mathbf{r}}$. We define the speed of a robot and the change of its heading that are affected by the simulated flow field as $s_{\mathsf{h}}^k = \|\dot{\mathbf{r}}\|$ and $\Delta\theta_k = \angle\dot{\mathbf{r}} = \arctan\left(\dot{r}_y/\dot{r}_x\right)$, respectively. Then, with time step size Δt, we rotate each robot by $\Delta\theta_k \Delta t$ and move them forward by $s_{\mathsf{h}}^k \Delta t$. We repeat this process until the length of the dead-reckoning trajectory of each robot reaches a predetermined travel distance D (i.e., $k s_{\mathsf{h}} \Delta t < D$). For our experiments, we used $c = \frac{3\sqrt{2}}{10}$, $\Delta t = 0.1$, $s_{\mathsf{h}} = 0.3\,\text{m/s}$, and $D = 1.4\,\text{m}$.

Algorithm 2: Nominal motion of a Khepera III robot

Data: Initial heading θ_0 of the robot
Set $k = 0$.
repeat
  Let $k = k + 1$.
  for $i = 1$ **to** 9 **do**
    $I_i \leftarrow$ the intensity of ambient light from the ith IR sensor
  end
  $I = \dfrac{\sum_i I_i}{9}$
  $\mathbf{f} = \dfrac{c(I_{\max} - I)}{I_{\max}} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$
  $\dot{\mathbf{r}} = s_{\mathsf{h}} \begin{bmatrix} \cos(\theta_0) \\ \sin(\theta_0) \end{bmatrix} + \mathbf{f}$
  $\Delta\theta_k = \arctan\left(\dfrac{\dot{r}_y}{\dot{r}_x}\right)$
  $s_{\mathsf{h}}^k = \|\dot{\mathbf{r}}\|$
  Rotate the robot by $\Delta\theta_k \Delta t$
  Move the robot forward by $s_{\mathsf{h}}^k \Delta t$
until $k < \dfrac{D}{s_{\mathsf{h}} \Delta t}$

We ran multiple sets of experiment using four Khepera III robots and chose ten navigation data sets of the robots—five from the right side of the domain to the left and five from the top to the bottom—shown in Fig. 3a. Given the collected navigation data sets, we reconstructed the simulated flow field from the trajectories and dead-reckoning errors of the robots by running the Glider CT algorithm (Algorithm 1). Because of the unknown trajectories of the robots, we assume the actual robot trajectories are straight lines between their starting and final positions. For the algorithm, the true field is unknown. We chose $\lambda^{(k,i)} = 0.01$ for all k, i where $k = \{1, 2, \ldots\}$, $i = \{1, \ldots, m\}$, and for the kth iteration, the iteration ended when both $\|\mathbf{f}_x^{(k,m)} - \mathbf{f}_x^{(k,0)}\|$ and $\|\mathbf{f}_y^{(k,m)} - \mathbf{f}_y^{(k,0)}\|$ are less than 10^{-3}. Figure 3b shows the reconstructed flow $\mathbf{f}^{\mathsf{reconst}}$. Compared to the true field $\mathbf{f}^{\mathsf{true}}$ in Fig. 2b, the reconstructed field suffers from noise. The magnitude of flow in the true field ranges

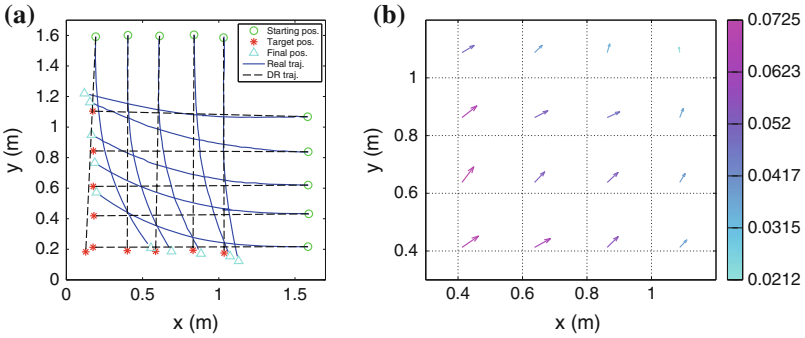


Fig. 3 Experimental results. **a** Trajectories of Khepera III robots. The *green circles*, *red stars*, and *cyan triangles* represent the starting positions, target positions, and ending positions of the robots, respectively. *Solid lines* connecting starting positions and ending positions are real trajectories, and *dashed lines* connecting starting positions and target positions are dead-reckoning trajectories. **b** The 4×4 flow field is reconstructed from navigation data of Khepera III robots shown in (**a**)

from 0.028 to 0.119 m/s, and that in the reconstructed field ranges from 0.0212 to 0.0725 m/s. We compute the error between the true and reconstructed fields by $\mathbf{e} = \mathbf{f}^{\text{true}} - \mathbf{f}^{\text{reconst}}$. The root-mean-square errors in the x and y components are $\mathbf{e}_x^{rms} = 0.0182$ m/s and $\mathbf{e}_y^{rms} = 0.0169$ m/s, respectively. We analyze that the error is partially due to the limitation of motor control for the differential wheels of Khepera III robots. The motor is controlled by pulse signals, and one pulse signal sent to the motors of a Khepera III robot rotates the robot by 0.06°. That is, the rotation angle is a multiple of 0.06°. Accumulated errors along the trajectories by the limitation of motor control may significantly affect the reconstruction of the field.

7 Conclusion

Glider CT reconstructs a depth-averaged flow field from the dead-reckoning errors of gliders. The Glider CT algorithm is a row-action iterative numerical method that converges to the solution of a set of nonlinear system equations sequentially. This paper proved the convergence of the Glider CT algorithm and demonstrated the effectiveness of the algorithm through experiments, in which Khepera III imitated the horizontal motion of underwater gliders under a simulated flow field. The experimental results suggest that the Glider CT algorithm may be applied to real gliders in future ocean sensing deployments.

Acknowledgments The research work is supported by ONR grants N00014-09-1-1074 and N00014-10-10712 (YIP), and NSF grants ECCS-0841195 (CAREER), CNS-0931576, OCE-1032285, and IIS-1319874.

References

1. Censor, Y.: Row-action methods for huge and sparse systems and their applications. SIAM Rev. **23**(4), 444–466 (1981)
2. Chang, D., Liang, X., Wu, W., Edwards, C.R., Zhang, F.: Real-time modeling of ocean currents for navigating underwater glider sensing networks. In: Koubâa, A., Khelil, A. (eds.) Cooperative Robots and Sensor Networks, Studies in Computational Intelligence, vol. 507, pp. 61–75 Springer, Berlin, Heidelberg (2014)
3. Chang, D., Zhang, F., Edwards, C.R.: Real-time guidance of underwater gliders assisted by predictive ocean models. J. Atmos. Oceanic Technol. **32**(3), 562–578 (2015)
4. Fernández-Perdomo, E., Cabrera-Gámez, J., Hernández-Sosa, D., Isern-González, J., Domínguez-Brito, A.C., Redondo, A., Coca, J., Ramos, A.G., Fanjul, E.A., García, M.: Path planning for gliders using Regional Ocean Models: Application of Pinzón path planner with the ESEOAT model and the RU27 trans-Atlantic flight data. In: Proceedings of OCEANS 2010, pp. 1–10 (2010)
5. Leonard, N.E., Paley, D.A., Davis, R.E., Fratantoni, D.M., Lekien, F., Zhang, F.: Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay. J. Field Robot. **27**(6), 718–740 (2010)
6. Martínez, J.M., De Sampaio, R.J.: Parallel and sequential Kaczmarz methods for solving underdetermined nonlinear equations. J. Comput. Appl. Math. **15**(3), 311–321 (1986)
7. Meldrum, D.T., Haddrell, T.: GPS in autonomous underwater vehicles. In: Proceedings of the Sixth International Conference on Electronic Engineering in Oceanography, pp. 11–17 (1994)
8. Merckelbach, L.M., Briggs, R.D., Smeed, D.A., Griffiths, G.: Current measurements from autonomous underwater gliders. In: Proceedings of the IEEE/OES/CMTC Ninth Working Conference on Current Measurement Technology, pp. 61–67 (2008)
9. Meyn, K.H.: Solution of underdetermined nonlinear equations by stationary iteration methods. Numerische Mathematik **42**(2), 161–172 (1983)
10. Nicholson, J.W., Healey, A.J.: The present state of autonomous underwater vehicle (AUV) applications and technologies. Marine Technol. Soc. J. **42**(1), 44–51 (2008)
11. Paley, D.A., Zhang, F., Leonard, N.E.: Cooperative control for ocean sampling: the glider coordinated control system. IEEE Trans. Control Syst. Technol. **16**(4), 735–744 (2008)
12. Penrose, R.: A generalized inverse for matrices. Math. Proc. Cambridge Philos. Soc. **51**(3), 406–413 (1955)
13. Schofield, O., Kohut, J., Aragon, D., Creed, L., Graver, J., Haldeman, C., Kerfoot, J., Roarty, H., Jones, C.P., Webb, D., Glenn, S.: Slocum gliders: robust and ready. J. Field Robot. **24**(6), 473–485 (2007)
14. Schomberg, H.: An improved approach to reconstructive ultrasound tomography. J. Phys. D: Appl. Phys. **11**(15), L181–L185 (1978)
15. Smith, R.N., Pereira, A.A., Chao, Y., Li, P.P., Caron, D.A., Jones, B.H., Sukhatme, G.S.: Autonomous underwater vehicle trajectory design coupled with predictive ocean models: a case study. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation, pp. 4770–4777 (2010)
16. Wu, W., Chang, D., Zhang, F.: Glider CT: Reconstructing flow fields from predicted motion of underwater gliders. In: The Eighth ACM International Conference on Underwater Networks and Systems, p. 47 (2013)
17. Zhang, F., Fratantoni, D.M., Paley, D.A., Lund, J.M., Leonard, N.E.: Control of coordinated patterns for ocean sampling. Int. J. Control **80**(7), 1186–1199 (2007)

Path Planning for Multi-agent Jellyfish Removal Robot System JEROS and Experimental Tests

Donghoon Kim, Hanguen Kim, Hyungjin Kim, Jae-Uk Shin, Hyun Myung and Young-Geun Kim

Abstract Over the recent years, the increasing influence of climate change has given rise to an uncontrolled proliferation of jellyfish in marine habitats, which has visibly damaged many ecosystems, industries, and human health. To resolve this issue, our team developed a robotic system to successfully and efficiently remove jellyfishes, named JEROS (Jellyfish Elimination RObotic Swarm). The JEROS consists of multiple USVs (Unmanned Surface Vehicles) that freely move in a marine environment to scavenge for and eliminate jellyfishes. In this paper, we propose a constrained formation control algorithm that enhances the efficiency of jellyfish removal. Our formation control algorithm is designed in consideration of the characteristic features of JEROS. It is designed to effectively work with the simple leader-follower algorithm. The leader-follower formation control does not work well if a reference path of the leader is generated without considering a minimum turning radius. In order to overcome such a limitation, a new path planning method—angular rate-constrained path planning—is proposed in this paper. The performance of the jellyfish removal function was tested at Masan Bay in the Southern coast of South Korea and formation control tests were conducted at Bang-dong Reservoir in Daejeon, South Korea.

D. Kim · H. Kim · H. Kim · J.-U. Shin · H. Myung (✉)
URL (Urban Robotics Lab), KAIST, 291 Daehak-ro, Yuseong-gu,
Daejeon 305-701, South Korea
e-mail: hmyung@kaist.ac.kr

D. Kim
e-mail: dh8607@kaist.ac.kr

H. Kim
e-mail: sskhk05@kaist.ac.kr

H. Kim
e-mail: hjkim86@kaist.ac.kr

J.-U. Shin
e-mail: jacksju@kaist.ac.kr

Y.-G. Kim
#303 Daejeon Intelligent Robot Engineering Center, Rastech, Inc,
35, Techno 9-ro, Yuseong-gu, Daejeon 305-510, South Korea
e-mail: kimyg@rastech.co.kr

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_21

Keywords Jellyfish removal robot · Unamanned surface vehicle · Path planning · Multi-agent robot

1 Introduction

Recently, the proliferation of jellyfish has emerged as a serious environmental issue that has threatened marine ecosystems and caused an enormous damage to marine-related industries in more than 14 countries around the world. In South Korea, the overall financial damage to marine-related industries was estimated to be over 300 million USD per year in 2009 [2]. In particular, the fishery industries, seaside power plants, and oceanic tourism enterprises have taken the most serious hit. The most prevalent species of jellyfish along the coast of South Korea are *Aurelia aurita* and *Nemopilema nomurai*. Some jellyfish species such as *Nemopilema nomurai* even have strong venom that can lead people to death. To cope with this problem, a number of studies pertaining to jellyfish removal have been actively performed. A previous study has developed a system consisting of two trawl boats equipped with jellyfish cutting nets [5, 13]. Utilizing large ships and many human operators, the system has shown high performance in jellyfish removal, but was limited in terms of its difficulty to operate in narrow and shallow coastal areas. In addition, numerous other systems were developed for the purpose of preventing the influx of jellyfish into water intake pipes of power plants. One of these systems consisted of a camera and a water pump [11]. Yet another system utilized a bubble generator and a conveyor device [10]. However, these types of systems are very expensive to install and maintain. To provide a cost-effective solution to the jellyfish problem, an earlier version of the autonomous jellyfish removal robot system, named JEROS (Jellyfish Elimination RObotic Swarm) consisting of a USV (Unmanned Surface Vehicle) part and a jellyfish remover part, was presented in [7]. The USV is designed based on a twin-hull-type ship that is stable to external disturbances, and the remover part shreds jellyfish using a rapidly rotating blade. An electrical control system for autonomous navigation is embedded in the prototype of JEROS. The design of the ship, navigation and image processing algorithms, and feasibility tests for the algorithms and jellyfish removal were introduced.

In this paper, the enhanced design of mechanical and electrical systems and the multi-agent robot system of JEROS are presented. The robot system is extended to a multi-agent robot system composed of three USVs to enhance the efficiency of jellyfish removal, and the leader-follower scheme [1] is employed and enhanced to control the formation of multiple robots. In the enhanced scheme, each follower robot follows not only its desired position for formation control, but also the speed and heading angle of the leader robot by utilizing the line-of-sight (LOS) guidance algorithm [4]. Additionally, for the autonomous navigation functionality considering formation control of the multi-agent robot system, a novel angular rate-constrained path planning algorithm based on the Theta* path planning algorithm [12] is introduced. Since the minimal turning radius and backward steering of JEROS are limited,

the small turning radius or backward input to follower robots caused by sharp turn of leader robot can lead to a large formation error. The proposed algorithm generates a smooth path considering these constraints. Finally, we carried out field tests at Bang-dong Reservoir in Daejeon, South Korea to demonstrate the enhancement in the performance of formation control by using a path generated by the proposed path planning algorithm. The result of formation control with the generated path using the proposed path planning algorithm is compared with those using A* and Theta*. The performance of the jellyfish removal was also demonstrated through field tests in Masan Bay located in South Korea.

The paper is organized as follows: in Sect. 2, the design and implementation of JEROS, the formation control based on the leader-follower scheme, and the novel path planning algorithm based on Theta* are described; in Sect. 3, experimental results of field tests for formation control and jellyfish removal are presented. Finally, in Sect. 4, we summarize this paper and discuss future works.

2 Formation Control and Path Planning of JEROS

2.1 Design of JEROS

The JEROS is composed of two parts as outlined in Fig. 1. One is the USV part, and the other is the remover part. The USV part can be operated alone, independent of the attachment of the remover part. In the remover part, a funnel-shaped net guides jellyfish to be gathered near the blade. The total mass of the USV part of JEROS weighs about 45 kg. Through some experiments at a fresh water tank, it is found that the maximum payload of USV is about 40 kg for the robot to remain stable in sea-state 2. The sea-state is the condition index of sea with respect to wind and wave,

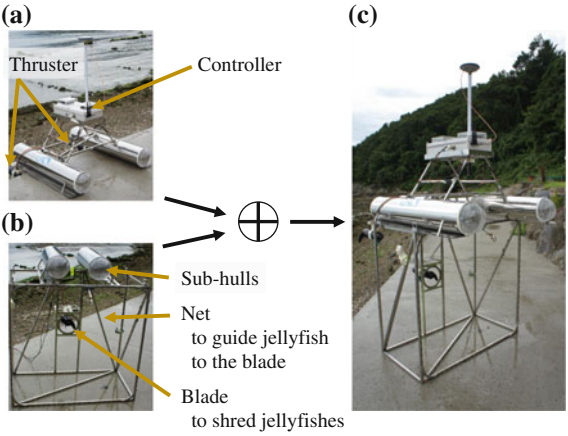


Fig. 1 JEROS consisting of a USV part and a remover part. **a** USV part. **b** Removal part. **c** JEROS

Table 1 Components used in controller

Component	Model	Manufacturer
GPS	OEM Star	Novatel
IMU	EBIMU-9DOF	E2BOX
Global network	3G Modem	Qualcomm
Local network	EZBee-M100-EXT (Zigbee)	Chipsen
Computer	Core i7 SBC	Intel
Microprocessor	TMS320F2808	Texas Instrument
Thruster	Endura C2	Minnkota

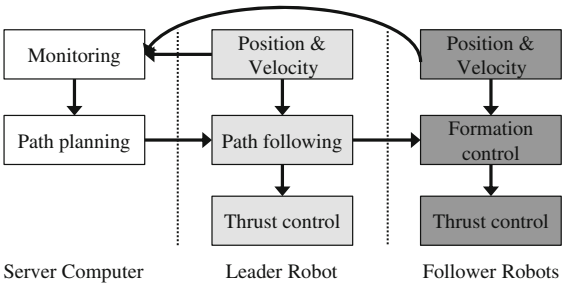


Fig. 2 Control scheme of JEROS

and sea-state 2 is characterized as smooth wave. The payload is increased at the sea due to the larger specific gravity of seawater than that of fresh water. The remover part can float by itself by virtue of the sub-hulls. The USV is operated as a differential drive robot with two thrusters installed on the rear portion of the USV. The tested maximum speed is about 2m/s. The USV is designed as a twin-hull-type since it is more stable against waves than mono-hull-type and easy to increase the payload using large hull.

The components used to make the controller are listed in Table 1. The position and heading angle are measured by GPS and IMU. The high-level control algorithm such as path-planning and formation control is computed by the SBC (Single Board Computer), and the low-level control such as controlling the thrusts is performed by the DSP. The control scheme is illustrated in Fig. 2. For communications between the USVs and the server computer at a remote place, two kinds of networks are used. To monitor and operate manually, the 3rd Generation mobile network (3G) is used. For local communications between the USVs, the ZigBee wireless network is used.

2.2 Formation Control

In order to overcome the limited area coverage of a single robot system, we employ a multi-agent cooperation system. The application of a multi-agent robot system

enhances the jellyfish removal capacity that is commensurate with the number of robot systems used.

For the formation control of the multi-agent robot system, the leader-follower method is employed due to its simplicity and low computation time. According to the intended scheme of the leader-follower method, the leader robot follows the reference path generated from the path planning algorithm using LOS guidance law, whereas the follower receives the location and velocity information of the leader robot and maintains the formation. The follower robot, for maintaining the formation, creates a waypoint based on the information of the leader robot, and is controlled to follow the waypoint. In order to properly perform formation control, it is imperative to accurately pursue the waypoint's location, heading angle, and the target speed at the location, but in marine environment faced with numerous disturbances, making the robot follow correct positions is not an easy task. In order to solve this problem, the follower robot is controlled as simply controlling a target its heading angle and a speed using the LOS guidance law in our system. The LOS guidance law provides target heading angle to asymptotically follow the position and target heading angle of the waypoint, which is case 1 as shown in Fig. 3. However, if the distance from follower robot to the waypoint is larger than R, the location of the waypoint is followed, which is the case 2 and 3 as shown in Fig. 3. The speed is controlled proportional to the distance from the follower robot to the waypoint.

The position of each robot is described by $\mathbf{X}_i = (x_i, y_i, \Psi_i)$ where x_i, y_i, and Ψ_i denote x, y coordinates, and heading angle of the robot, respectively. Ψ_i^* indicates target heading angle. The position of the leader and follower robots are denoted by $\mathbf{X}_l$, $\mathbf{X}_f$, and the waypoint which is the desired position of the follower robot is

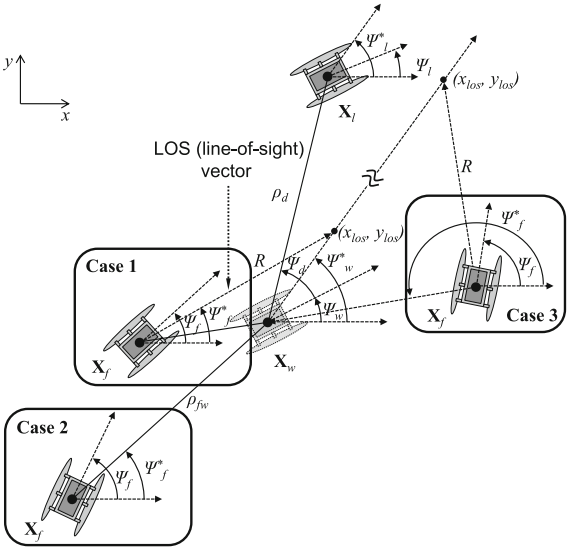


Fig. 3 Overview of the proposed formation control algorithm and explanation of notations

denoted by $\mathbf{X}_w$, as shown in Fig. 3. $\mathbf{X}_w$ is determined by the desired displacement and heading angle from $\mathbf{X}_l$, and is calculated as follows:

$$\mathbf{X}_w = \begin{pmatrix} x_l - \rho_d \cos(\Psi_d + \Psi_w) \\ y_l - \rho_d \sin(\Psi_d + \Psi_w) \\ \Psi_l \end{pmatrix}, \tag{1}$$

where ρ_d and Ψ_d denote the desired displacement and heading angle. Ψ_w and Ψ_w^* should be Ψ_l and Ψ_l^*, respectively. The desired speed of a follower robot at the location of the waypoint, v_w, is determined with consideration of the yaw rate of the leader robot, $\dot{\Psi}_l$, and the displacement of the follower robot as follows:

$$v_w = v_l + \rho_d \sin(\Psi_d)\, \dot{\Psi}_l. \tag{2}$$

The target heading angle of the follower robot, Ψ_f^*, is calculated as follows:

$$\Psi_f^* = \begin{cases} \tan^{-1}\left(\dfrac{y_{los} - y_f}{x_{los} - x_f}\right) & 0 \le \rho_c < R + k_\alpha v_w \ \text{(case 1)} \\ \tan^{-1}\left(\dfrac{y_w - y_f}{x_w - x_f}\right) & \text{otherwise (case 2, 3)} \end{cases}, \tag{3}$$

where k_α is a constant gain. ρ_c is the distance between $\mathbf{X}_w$ and the LOS point, (x_{los}, y_{los}), in the direction of Ψ_w^*. In case 1 as shown in Fig. 3, the follower robot follows the waypoint and the leader robot's target heading angle by tracking a virtual path asymptotically using the LOS guidance law. The virtual path means a straight line from $\mathbf{X}_w$ in direction of Ψ_w^*. The LOS vector is a vector from $\mathbf{X}_f$ to the LOS point and its length is constant, R. The LOS point is a target point to track the virtual path. The case 2 and 3 indicate the large error state of the follower robot in the conditions of $\rho_c < 0$ and $\rho_c > R + k_\alpha v_w$, respectively. In these cases, Ψ_f^* is the angle from the follower robot to the waypoint and is calculated as shown in the second row of (3).

The target speed of the follower robot, v_f^*, is described as follows:

$$v_f^* = \begin{cases} v_{f,\max} - \dfrac{(v_{f,\max} - v_w)\,\rho_c}{R} & 0 \le \rho_c < R \\ v_w - \dfrac{v_w(\rho_c - R)}{k_\alpha v_w} & R \le \rho_c < R + k_\alpha v_w \\ \dfrac{v_{f,\max}(\rho_c - R + k_\alpha v_w)}{k_\alpha v_w} & R + k_\alpha v_w \le \rho_c \\ & \text{and } \rho_c < R + 2k_\alpha v_w \\ v_{f,\max} & \text{otherwise} \end{cases}, \tag{4}$$

where $v_f^* \in [0, v_{f,\max})$ and $v_{f,\max}$ is the maximum speed of the follower robot. Equation (4) indicates a speed profile for the follower robot's waypoint tracking, which is designed as a linear interpolation with four inflection points with respect to ρ_c.

The points are $\left(0, v_{f,\max}\right)$, (R, v_w), $(R + k_\alpha v_w, 0)$, $\left(R + 2k_\alpha v_w, v_{f,\max}\right)$. When the follower robot reaches near the waypoint, i.e., $\rho_c \simeq R$, v_f^* converges to v_w. If the robot reaches near the point of $\rho_c = R + k_\alpha v_w$ beyond the waypoint, v_f^* converges to 0. If $\rho_c > R + k_\alpha v_w$, the follower goes behind itself toward X_w with calculated speed. Otherwise, the follower heads for $\mathbf{X}_w$ directly with maximum speed.

Since the leader-follower formation control does not consider the information of the follower robot in generating waypoints, the path which the follower cannot follow can be generated. For example, the waypoint behind the follower can cause a serious problem. Therefore, the constraints such as the minimum turning radius need to be considered when the leader's path is generated.

2.3 Path Planning and Following

Based on [6, 8], we propose a new approach, named Angular Rate-Constrained Theta* (ARC-Theta*), to create paths with considerations of the formation state and the vehicle's performance. The proposed algorithm in this study is based on Theta*, which is similar to A* [12]. The basic Theta* selects a parent node by checking the LOS (Line of Sight). When connecting between the current search node s and its parent node $s.parent$, Theta* always checks the LOS, enabling it to search its neighbors $s_{neighbor}$ in any direction within a specified distance. One of the key concepts used in the proposed algorithm is to restrict the range of LOS and the angular rate by accommodating for the turning ability of the USV formation. Thus, the angular rate of each LOS is calculated when the node is expanded in Theta*. The angular rate r is defined as the ratio of the leader's speed V to the turning radius T_{total}, that is calculated from the leader's turning radius T_{leader} and the distance ρ_{lf} between the leader and the follower for a USV formation as follows:

$$\begin{aligned} r &= \frac{V}{T_{total}} \\ T_{total} &= T_{leader} + \rho_{lf}. \end{aligned} \tag{5}$$

Figure 4 shows an example of the modified LOS, named as Angular Rate-Constrained LOS (ARC-LOS) that reflects the constraints. The ARC-LOS function always checks whether or not the current angular rate is greater than the maximum angular rate. If the current angular rate is greater than the maximum angular rate, the ARC-LOS function returns a false state. A function, *IsWalakble* checks the occupancy states around the current search node s on the weighted occupancy grid map. The occupancy states are calculated according to the vehicle orientation and size. If LOS is ensured, the ARC-LOS function returns the average occupancy cost according to the distance. Even so, the method cannot satisfy the arrival heading angle at the goal point because the ARC-LOS algorithm only verifies LOS between current and previous nodes (explored node and its parent node). To mitigate this problem, Dubin's curve algorithm is applied at the start point and the goal point. Dubin's curve algorithm is used to get an optimal path under the rule that the vehicle turns

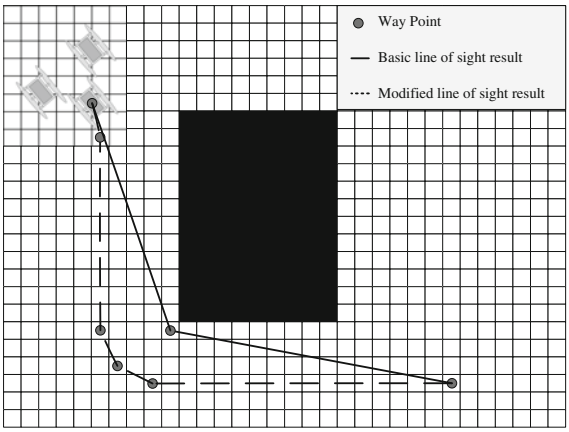


Fig. 4 Example of angular rate-constrained LOS result

left or right when the angular rate of the vehicle is given, with the assumption that the vehicle cannot reverse [9]. To apply Dubin's curve algorithm at the start and goal points, at first, a path from the start point to the goal point is created by Theta* with the ARC-LOS function. Then additional way-points are calculated using the Dubin's curve algorithm. This is possible because Theta* creates way-points using LOS so that the LOS between any two contiguous way-points is ensured. Thus, the influence of obstacles on the path can be avoided by creating Dubin's curve except in cases where the start angle or goal angle do not satisfy the LOS condition. If obstacles exist on the path generated with maximum angular rate at the start and goal points, the problem can be solved by decreasing the angular rate and re-calculating Dubin's curve. The pseudo code of the proposed path planning algorithm is shown in Algorithm 1. A function, *CreateDubinsCurves* generates the Dubin's curve at the start point and goal point after applying Theta* with the ARC-LOS function.

Algorithm 1 ARC-Theta*(s_{start}, s_{goal})

```
1: s_start.Parent ← s_start
2: while open ≠ ∅ do
3:    s ← open.Pop()
4:    for each s_neighbor do
5:       if AngularRateConstrained LOS(s_neighbor, s) then
6:          s_neighbor.parent ← parent
7:          open.Push(s_neighbor)
8:       end if
9:    end for
10: end while
11: if CreateDubinsCurves(s_start, s_goal) then
12:    return path found
13: else
14:    return no path found
15: end if
```

The navigation system calculates desired heading angles to follow given paths using the LOS guidance algorithm. The LOS guidance algorithm computes the LOS vector to calculate a control input to steer the vehicle. The LOS vector is formed by connecting the robot position to an intersecting point on the path at a distance of the tracking radius ahead of the robot.

3 Experiments

3.1 Formation Control Tests

Field tests were conducted at Bang-dong Reservoir, Daejeon in South Korea to evaluate the feasibility of the formation control. Our multi-agent system consists of three JEROS prototypes (1 leader robot, 2 follower robots). Each robot is equipped with a GPS receiver (Novatel OEM-Star), which has 1.5 m accuracy, and an IMU (EBIMU), which measures the robot's position, heading angle, and velocity, for conducting localization. The leader robot followed the paths of A*, Theta*, and the proposed path, while the two follower robots generated waypoints from the information received from the leader robot and followed those paths. Both lengths of LoS vector for path following guidance and formation control were set to 3 m. The desired displacements and heading angles were set to $(\rho_{d1}, \Psi_{d1}) = (4, \pi/3)$ and $(\rho_{d2}, \Psi_{d2}) = (4, -\pi/3)$, respectively, which represented an equilateral triangle formation. The target speed of the leader robot v_l^* and the maximum speed of the follower robot $v_{f,\max}$ were set to 0.9 and 1.8 m/s, respectively.

Figure 5 shows the generated paths from A*, Theta*, and the proposed algorithm. To create the Zig-Zag paths in a specific area, we have assumed that the virtual

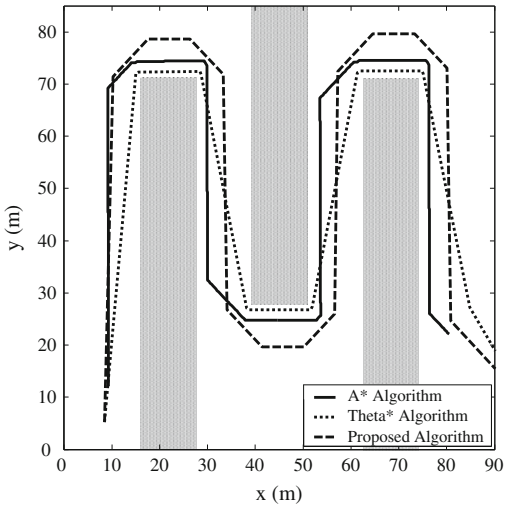


Fig. 5 Obstacle map denoted by *dotted* pattern and paths from A*, Theta*, and the proposed algorithms

Table 2 RMSE of the follower robots' position

	Follower robot 1 (m)	Follower robot 2 (m)
A*	1.01	0.97
Theta*	1.01	1.00
Proposed	0.83	0.72

obstacles were installed. A* and Theta* generate rapidly changing curves, in which some waypoints were generated behind the follower. However, it is clear that the proposed path, by virtue of its formation control function, successfully incorporated the restrictions in the turning radius, hence the previously observed problems did not occur. In the case of A*, the zig-zag patterns shown on the straight path due to map resolution caused regular occurrence of large errors, even when the leader was moving forward. In the case of Theta*, the generated path is composed of the waypoints on the inflection points. There were few errors when the robot was moving forward, but the event of a sudden curve resulted in a high error. The proposed path maintained a stable formation in comparison to A* and Theta*, and there was little error. The RMSE (Root-mean-squared error) results are listed in Table 2.

3.2 Jellyfish Removal Tests

The performance of the jellyfish removal function was tested at Masan Bay located in southern coast of South Korea. When the robot moved, the jellyfish swimming in the periphery of the robot's path were guided to the shredding blade by a net installed on the remover part. Then, the jellyfishes were cut into small pieces as shown in Fig. 6. In this experiment, it was verified that JEROS can remove approximately 36 *Aurelia aurita* jellyfish per 1 min (approximately 3.6 kg) when the speed of JEROS is 0.5 m/s and the entrance area of the remover part 1.44 m^2. Figure 6 visually shows the shredded remnants of many jellyfishes caught in the cutting blades.

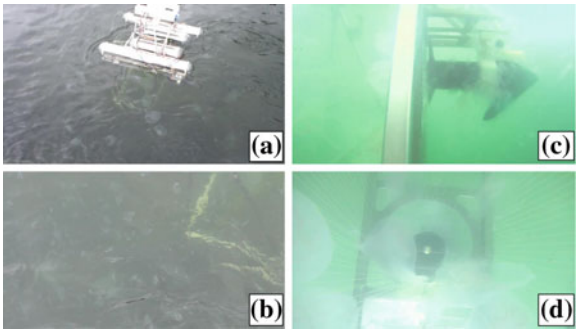


Fig. 6 Jellyfish removal test: **a** JEROS on the water **b** image of small pieces of shredded jellyfishes **c** image of small pieces of shredded jellyfishes in taken underwater **d** the jellyfishes guided to the blade

4 Conclusions

In this paper, we presented the leader-follower scheme-based formation control and a novel path planning algorithm based on Theta* for the multi-agent autonomous jellyfish removal robot system, JEROS. In addition, field tests were performed in order to assess the performance of formation control and jellyfish removal functionality. To enhance the performance of jellyfish removal, the robot system was modified compared to the previous version with respect to its dimensions, thrust force, and wireless communication method; and it was extended to a multi-agent robot system composed of three prototypes of JEROS. To accomplish the autonomous navigation of the multi-agent robot system, a leader-follower scheme was employed to control their formation. A novel path planning algorithm based on Theta* was employed to plan a path considering the formation state and the vehicle's constraints. The performance of the formation control was demonstrated through field tests in a reservoir in South Korea. The result of the test using a path generated by the proposed path planning algorithm showed smooth and stable formation control compared with the results using A* and Theta*. Additionally, the performance of jellyfish removal was estimated to be about 3.6 kg/min, on average, through field tests at Masan Bay located in South Korea. Future research will be focused on creating more advanced formation control algorithms and conducting more elaborate investigations of the efficiency of JEROS through various field tests.

Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A1A05011746). Mr. H. Kim and Mr. H. Kim are supported by Korea Ministry of Land, Transport and Maritime Affairs (MLTM) as U-City Master and Doctor Course Grant Program.

References

1. Breivik, M., Hovstein, V.E., Fossen, T.I.: Ship formation control: a guided leader-follower approach. In: IFAC World Congress (2008)
2. Choi, H.-S.: Scientists seek beneficial uses for jellyfish. The Korea Herald. http://www.koreaherald.com/view.php?ud=20120826000052 (2012). Accessed 17 June 2014
3. Dunbabin, M., Lang, B., Wood, B.: Vision-based docking using an autonomous surface vehicle. In: IEEE International Conference on Robot and Automation (ICRA) (2008)
4. Fossen, T.: Marine control systems: guidance, navigation and control of ships, rigs and underwater vehicles. Mar. Cybern. (2002)
5. Kim, I.-O., An, H.-C., Shin, J.-K., Cha, B.-J.: The development of basic structure of jellyfish separator system for a trawl net. J. Korean Soc. Fish Technol. **44**(2), 99–111 (2008). (in Korean with English abstract)
6. Kim, H., Lee, T., Chung, H., Son, N., Myung, H.: Any-angle path planning with limit-cycle circle set for marine surface vehicle. IEEEInternational Conference on Robot and Automation (ICRA) (2012)

7. Kim, D., Shin, J.-U., Kim, H., Kim, H., Lee, D., Lee, S.-M., Myung, H.: Design and Implementation of Unmanned Surface Vehicle JEROS for Jellyfish removal. J. Korea Robot. Soc. **8**(1), 51–57 (2013)
8. Kim, H., Kim, D., Shin, J.-U., Kim, H., Myung, H.: Angular rate-constrained path planning algorithm for unmanned surface vehicles. Ocean Eng. **84**, 37–44 (2014)
9. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
10. Lee, J.-H., Kim, D.-S., Lee, W.-J., Lee, S.-B.: System and method to prevent the impingement of marine organisms at the intake of power plants. Korean Patent 10-0558267-00-00 (2006)
11. Matsuura, F., Fujisawa, N., Ishikawa, S.: Detection and removal of jellyfish using underwater image analysis. J. Vis. **10**(3), 259–260 (2007)
12. Nash, A., Daniel, K., Koenig, S., Felner, A.: Theta*: any-angle path planning on grids. In: National Conference on Artificial Intelligence (AAAI) (2007)
13. NFRDI.: Trends of overseas fisheries. Technical Report 2 (National Fisheries Research and Development Institute (NFRDI) of South Korea issued in Korean) (2005)

Motion Planning of Multiple Mobile Robots Based on Artificial Potential for Human Behavior and Robot Congestion

Satoshi Hoshino and Koichiro Maki

Abstract In order for robots to exist together with humans, safety for humans has to be ensured. On the other hand, safety might decrease working efficiency of robots. Namely, this is a trade-off problem between the human safety and robot efficiency in a field of human-robot interaction. For this problem, we propose a novel motion planning technique of multiple mobile robots. Two artificial potentials are presented for generating repulsive force. A behavior potential is provided for humans. A congestion potential is provided for robots. Through simulation experiments, the effectiveness of the behavior and congestion potentials used in the motion planning technique for the human safety and robot efficiency is discussed. Moreover, a sensing system for humans in a real environment is developed. Finally, the significance of the potential generated from the actual human behavior is discussed.

Keywords Multi-robot systems · Motion planning · Artificial potential method · Human-robot interaction

1 Introduction

Nowadays, service robots, such as automated guided vehicles in factories, nursing and caring robots in hospitals and care facilities, and cleaning and guiding robots in offices play an important role. In such robot systems, human safety is a primary concern [1]. Next to the safety, working efficiency of robots is improved. In consideration of this practical issue, we focus on systems in which multiple autonomous mobile robots exist together with humans.

In the systems, working robots are required to avoid collisions with humans while moving toward a destination. This is so-called the motion planning of the robots. In this paper, an artificial potential method is basically applied [2]. The configuration space is defined by attractive and repulsive potential fields. In general, the repulsive

S. Hoshino (✉) · K. Maki
Department of Mechanical and Intelligent Engineering, Utsunomiya University,
7-1-2 Yoto, Utsunomiya 321-8585, Japan
e-mail: hosino@cc.utsunomiya-u.ac.jp

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_22

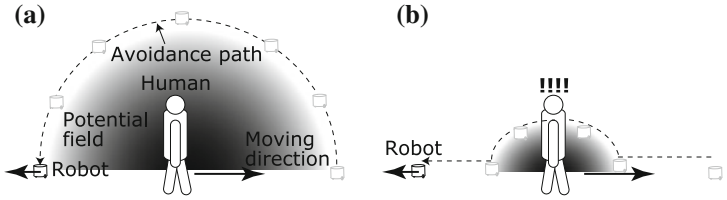


Fig. 1 Trade-off between safety and efficiency depending on the size of repulsive potential. **a** High safety versus low efficiency. **b** Low safety versus high efficiency

potential has been provided to the obstacle on the basis of the position. However, for moving obstacles such as humans, this approach is destined to cause a trade-off problem between the human safety and robot efficiency, as illustrated in Fig. 1.

If a robot performs avoidance motion based on a large potential for increasing human safety, working efficiency of the robot is decreased as the avoidance path lengthens (see Fig. 1a). On the other hand, as the avoidance path shortens by means of a small potential for increasing the working efficiency, collision risk due to a change in human behavior is increased (see Fig. 1b). In this case, the human safety is decreased. Moreover, robots that performing the avoidance motions based on the repulsive potential move in the same direction as illustrated in Fig. 2.

This swarm behavior results in a congestion of the robots. If other humans come to the congestion, the robots are not enabled to avoid the collisions with the human because of the others. Eventually, safety is not ensured. Furthermore, working efficiency of the robots is also decreased in the congested situation.

For the challenges, we present a repulsive potential that takes into account the behavior of obstacles in addition to the position. This is named as a behavior potential. For the robot congestion, we present another repulsive potential that takes into account the density of the robots. This is named as a congestion potential.

Through simulation experiments, the effectiveness of the behavior and congestion potentials used in the motion planning technique for the human safety and robot efficiency is discussed. Moreover, a sensing system for humans in a real environment is developed. Finally, the significance of the potential generated from the actual human behavior is discussed.

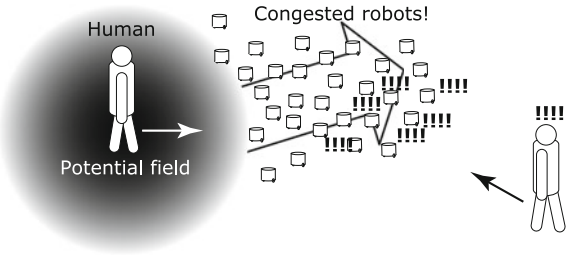


Fig. 2 Robot congestion caused by swarming behavior based on the same potential

2 Artificial Potential Method

2.1 Potential Field

Equation (1) expresses a basic model of the artificial potential method used in the motion planning. The attractive potential, U_{x_d}, allows robots to move toward a destination, x_d. The repulsive potential, U_o, allows robots to avoid collisions with an obstacle, o.

$$U = U_{x_d} + U_o \tag{1}$$

In this paper, the following attractive potential is provided in consideration of the gravitational energy: $U_{x_d}(x) = k_p(x - x_d)$, where k_p is a coefficient, x and x_d represent the positions of a robot and its destination. Hence, U_{x_d} affects the constant attractive force k_p on the robots as follows: $\boldsymbol{F}_{x_d} = -\nabla U_{x_d} = -k_p$.

For avoiding the collisions with obstacles, the following repulsive potential has been widely used [2]: $U_o(x) = \frac{1}{2}\eta\left(\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right)^2$, where η is a coefficient, $\rho(x)$ represents the distance from the robot position, x, to the center of the obstacle o, and ρ_0 defines the threshold. This means that the robot within the threshold, $\rho(x) \leq \rho_0$, is affected by the repulsive force as follows: $\boldsymbol{F}_o(x) = -\nabla U_o = \eta\left(\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(x)}\nabla\rho(x)$.

2.2 Approaches to Problems of Previous Repulsive Potential

The repulsive potential has been provided so as to uniformly affect the repulsive force on robots around an obstacle. However, actual robots have a limitation in the moving velocity. Additionally, such a uniform potential for moving obstacles, which are humans gives rise to the trade-off problem (see Fig. 1). For this problem, a more flexible potential, U_{o_h}, the shape of which varies according to the moving velocity and direction of the humans, in addition to the position, is required.

For each of the robots, a repulsive potential, u_{o_r}, is additionally provided. Hence, each robot is enabled to avoid collisions with other robots. In this regard, the total repulsive potential of the robots is derived as follows: $U_{o_r} = \sum u_{o_r}$. This means that the shape of the potential has multiple peaks. As a result, the robots are moved in the gradient directions from the different peaks and finally congested. Therefore, a repulsive potential with a single peak, U_{o_r}, is required.

For the problems described above, we provide behavior and congestion potentials for humans and robots. The approaches are as follows:

- Quantify the behavioral property of obstacles using the von Mises distribution;
- Identify the global robot congestion using Kernel density estimation (KDE).

The von Mises distribution is based on statistic, such as an angle and its dispersion on a circle. Therefore, it is capable of parameterizing the moving velocity and direction of an object. Thus, an existence probability of a human is given by the probability density function. This is used as the behavior potential U_{o_h}. The KDE is capable of extrapolating the probability density function of the existence probability for the entire space on the basis of the existence probability of each robot. The probability density function is identified as the global robot congestion and used as the congestion potential U_{o_r} with a single peak. Finally, the repulsive potential U_o in consideration of the behavioral property of humans and the global robot congestion is given by merging the potentials as follows: $U_{o_h} + U_{o_r}$.

3 Behavior Potential for Moving Obstacles

3.1 Related Works on Moving Obstacles

A local path planning method that changed the shape of the repulsive potential function has been proposed [3]. However, static obstacles were assumed in the literature. For moving obstacles, motion planning techniques based on the moving velocity and direction, in addition to the position, have been proposed [4, 5]. However, the shape of the repulsive potential fields were provided in the same way as previously [2]. In the potential fields, only the area where robots were affected by the repulsive force was defined depending on the behavioral property and position of the obstacle.

3.2 Mathematical Formulation of Behavior Potential

The behavior potential is capable of changing the shape on the basis of the moving velocity and direction of humans in addition to the position. Unlike the previous repulsive potential, a threshold of the distance between the robot and obstacle, ρ_0, is not defined. The behavior potential based on the von Mises distribution is provided as follows:

$$U_{o_h}(r, \theta) = \sum_{i=1}^{n} \frac{\exp\left[\kappa \cos(\theta_i - \mu_i)\right]}{2\pi I_0(\kappa)} \alpha_h \frac{\exp\left[-\dfrac{r_i}{2\sigma}\right]}{2\pi\sigma} \beta ||\boldsymbol{v}_i||, \tag{2}$$

where r represents the distance from the humans and θ defines the relative direction. κ is a measure of concentration on the moving direction derived from $SD \times ||\boldsymbol{v}||$. In this regard, SD is named as the statistical direction since it is defined as the inverse number of statistical variance of the human moving direction represented by μ. $\boldsymbol{v}$ represents the human moving velocity. σ is the variance of the radial component, α_h is the coefficient, and β is a coefficient of the velocity $\boldsymbol{v}$. $I_0(\kappa)$ represents the modified Bessel function of order 0.

From Eq. (2), the robots are enabled to define the repulsive potential for the moving obstacles depending on the relative positions, velocities, and directions. As a result, the potential affects the following repulsive force on the robots so that the robots avoid the collisions with the obstacles:

$$\boldsymbol{F}_{o_h}(r,\theta) = -\nabla U_{o_h}(r,\theta) = \sum_{i=1}^{n} \frac{-\alpha_h \beta ||\boldsymbol{v}_i||}{4\pi^2 \sigma I_0(\kappa)} \exp\left[\kappa \cos(\theta_i - \mu_i) - \frac{r_i}{2\sigma}\right] \boldsymbol{J}_{o_h}, \quad (3)$$

where $\boldsymbol{J}_{o_h}$ represents the following Jacobian:

$$\boldsymbol{J}_{o_h} = \begin{bmatrix} -\dfrac{\cos\theta}{2\sigma} + \dfrac{\kappa \sin\theta \sin(\theta-\mu)}{r} \\ -\dfrac{\sin\theta}{2\sigma} - \dfrac{\kappa \cos\theta \sin(\theta-\mu)}{r} \end{bmatrix}. \quad (4)$$

Figure 3 illustrates that a robot in the different potential fields is moving toward a destination while avoiding the collision with a human. The human moves from left to right linearly ignoring the robot. The attractive force moves the robot toward the destination. Therefore, the robot begins to avoid the human at the point where the repulsive force is greater than the attractive force.

In Fig. 3a, the repulsive potential with a circular shape is provided for the human from the following two-dimensional normal distribution function: $U_{o_h}(r) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{r^2}{2\sigma^2}\right]$. Although the repulsive force affects the robot, the robot keeps moving forward until the force is greater than the attractive one. Thus, the human and robot approaching each other result in the high collision risk. In addition, even if the robot is behind the human, it is affected by the repulsive force.

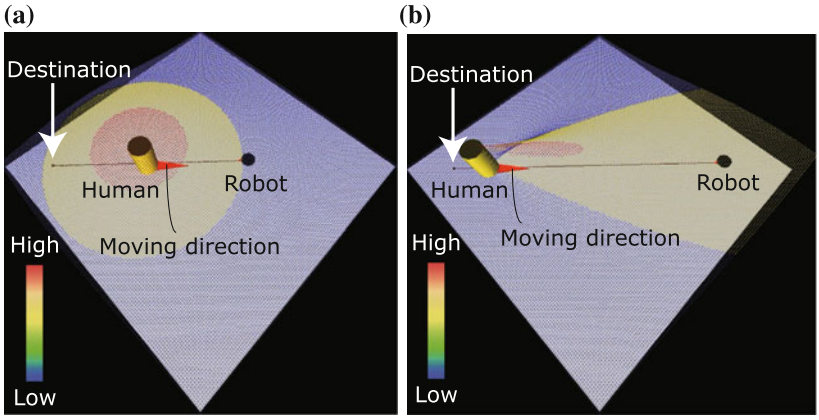


Fig. 3 Previous and proposed repulsive potential fields for moving obstacle. **a** Normal (uniform) potential. **b** Behavior potential

In Fig. 3b, Eq. (2) provides the behavior potential. In contrast to Fig. 3a, the shape of the potential is elongated in the moving direction. The volume of elongation is determined by the moving velocity. For the faster human, the robot is enabled to perform the avoidance motion from a distance. On other hand, for the human moving slowly, the robot performs the avoidance motion near the human. Furthermore, the robot behind the human is allowed to keep moving without any influence from the potential.

4 Congestion Potential for Robots

4.1 Related Works on Congestion

The first author, Hoshino, has shown that traffic jam of robots in the same straight line was solved by controlling the velocity [6]. For the robots moving in a two-dimensional surface, the moving direction is also useful for solving the jam [7]. However, if a team of robots performed the same congestion avoidance motion, the robots were congested in another area, while the current congestion was reduced. In recently, the global crowd has been calculated on the basis of a cellular automaton model [8]. However, the calculated result depended on the number of cells. Focusing on continuous systems, the congestion models based on the hydrodynamics have been proposed [9, 10]. However, while the detailed and global congestion was derivable, the fluid model required a huge amount of calculation.

4.2 Mathematical Formulation of Congestion Potential

For the problems described above, the KDE, Kernel density estimation, is used to identify the global robot congestion. Thus, each robot is enabled to know the density of the position on the basis of the global congestion. Finally, the congestion potential with a single peak is provided as follows:

$$U_{o_r}(r) = \frac{\alpha_r}{nb^2} \sum_{i=1}^{n} \frac{1}{2\pi} \exp\left[-\frac{r_i^2}{2b^2}\right], \tag{5}$$

where r represents the distance from the robots. α_r is a coefficient and b is a smoothing parameter determining the width of the peak in the potential.

Since the density of each robot is derived from the relative position, the total calculation cost for n robots is equal to $n(n-1)$. From Eq. (5), the potential with the single peak affects the following repulsive force on the robots so that the robots occupy the system uniformly:

$$\boldsymbol{F}_{o_r}(r) = -\nabla U_{o_r}(r) = \frac{\alpha_r \boldsymbol{J}_{o_r}}{2\pi n b^2} \sum_{i=1}^{n} \frac{r_i}{b^2} \exp\left[-\frac{r_i^2}{2b^2}\right], \tag{6}$$

where $\boldsymbol{J}_{o_r}$ is the following Jacobian based on the relative direction θ among the robots:

$$\boldsymbol{J}_{o_r} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}. \tag{7}$$

Figure 4 illustrates that two different potentials are provided to robots for the congestion. The robots are moving toward the center of the square. Note that it is not necessary to calculate the density of other points excepting the robot positions. The potentials are shown for illustrative purposes. In this paper, the robots are assumed to have uniform performance on the motion. From the assumptions, the following two-dimensional normal distribution function based only on the position is provided to each robot for the collision avoidance among the robots: $u_{o_r}(r) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{r^2}{2\sigma^2}\right]$.

In Fig. 4a, multiple peaks are generated in the potential field. Since each robot has the individual potential, u_{o_r}, all the potentials are combined in a simple manner as follows: $U_{o_r} = \sum u_{o_r}$. In this potential field, for the robots in adjacent peaks, different repulsive forces are affected in opposite direction. In consequence, the robots are newly congested at the trough of the potential.

In Fig. 4b, Eq. (5) provides the congestion potential for the robots. In contrast to Fig. 4a, a single peak is generated in the potential field by estimating the global robot congestion. In this potential, the repulsive force is affected on the robots in the gradient direction from the same peak. The robots are, therefore, enabled to perform the globally-coordinated motions for the congestion reduction.

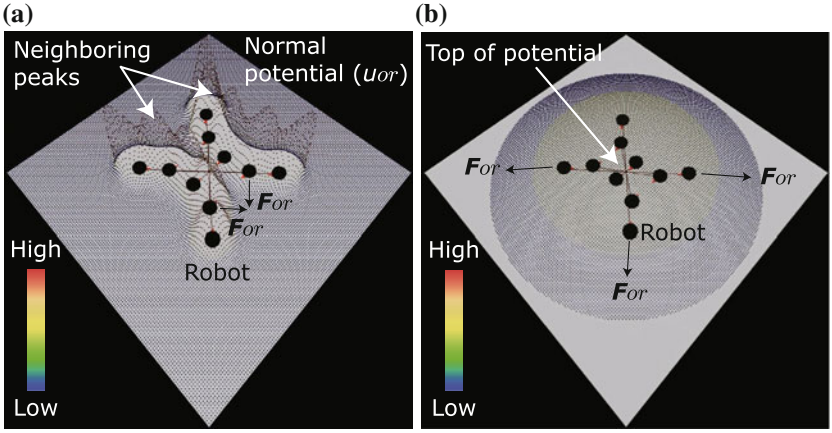


Fig. 4 Previous and proposed repulsive potential fields for robot congestion. **a** Multiple peaks generated from individual normal potentials u_{o_r}. **b** Single peak of globally estimated congestion potential U_{o_r}

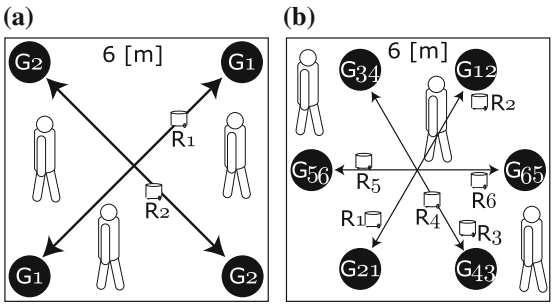


Fig. 5 Three humans and two/six robots are used in simulated rooms with $6 \times 6m^2$. G_i, G_{ij}, and G_{ji} in (**a**) and (**b**) represent destinations of robots R_i, R_{ij}, and R_{ji}. Inattentive person that moves directly to randomly-given destinations ignoring the robots is assumed. **a** Uncrowded room. **b** Crowded room

5 Simulation Experiments

5.1 Experimental Conditions

In this simulation experiment, holonomic robots are used as depicted in Fig. 5. For the humans, the behavior potential U_{o_h} based on Eq. (2) is provided. In addition, a two-dimensional normal potential and social potential field model presented in [11] are applied. For the robots, the normal potential u_{o_r} is provided for the collision avoidance. Moreover, the congestion potential U_{o_r} given by Eq. (5) is provided.

The maximum velocities of the humans and robots are given as 0.8 and 0.4 [m/s], respectively. Parameters in Eqs. (2) and (5) for the behavior and congestion potentials are: $\kappa = SD \times ||\boldsymbol{v}||$, $SD = 5.0, \sigma = 1.5, \alpha_h = 10000, \beta = 1.0, \alpha_r = 2000$, and $b = 4.0$. The robots are required to avoid collisions even with inattentive person such as children. Therefore, three humans are assumed to go straight to their randomly-given destinations without avoiding the robots.

As an index of safety, we focus on the number of collisions between the robots and humans. As for the robot efficiency, we measure the time before the total number of arrivals to their destinations is 100. Furthermore, the time and collision, which are the trade-off relationship, are multiplied together and called as T-O (abbr. of trade-off) value. The lower value indicates higher effectiveness of the potential.

5.2 Experimental Results and Effectiveness of Motion Planning

Tables 1 and 2 show the averaged results of five simulations for the combinations of potentials. NP, SPF, and BP represent the normal potential, social potential field model, and behavior potential. CP represents the congestion potential. Hereafter, the combination of the potentials is expressed by the form of U_{o_h}–U_{o_r}.

Table 1 Result for environment shown in Fig. 5a

U_o	U_{o_h}	NP	SPF	**BP**	NP
	U_{o_r}	NP			**CP**
Time [s]		736.7	417.3	476.6	586.7
Collisions		202.4	181	18.4	187.4
Time×Collisions		148,672	75,531	8,769	109,582

Table 2 Result for environment shown in Fig. 5b

U_o	U_{o_h}	NP	SPF	**BP**	NP
	U_{o_r}	NP			**CP**
Time [s]		228.5	128.2	154.0	230.6
Collisions		71.8	60.4	6.1	67.4
Time×Collisions		16,406	7,743	934	15,542

Table 3 Proposed potential U_o: U_{o_h} is "BP" and U_{o_r} is "CP"

	Time [s]	Collisions	Time×Collisions
Figure 5a	434.3	15.3	6,645
Figure 5b	149.8	4.6	689

In Table 1, NP–NP resulted in the longest time and most collisions when the U_{o_r} was NP. While SPF–NP resulted in the shortest time, the number of collisions was increased compared to that of BP–NP. Moreover, BP resulted in the lowest T-O value in the U_{o_h}. When U_{o_h} was NP, NP–CP reduced both the time and collisions compared to those of NP–NP. As a result, the lower T-O value was obtained by CP.

In Table 2, even in the crowded room, NP–NP resulted in the longest time and most collisions. While SPF–NP resulted in the shortest time, the number of collisions was increased compared to that of BP–NP. As for the T-O values, BP was the lowest in the U_{o_h}. On the other hand, NP–CP did not reduce the time compared to that of NP–NP. However, the number of collisions was reduced. Consequently, CP resulted in the lower T-O value in the U_{o_r}.

From these results, the behavior and congestion potentials were combined. The repulsive potential given by $U_o = U_{o_h} + U_{o_r}$ was provided for both the humans and robots in Fig. 5a, b. The simulation result is shown in Table 3.

Compared to the results shown in Tables 1 and 2, time was shortened and the number of collisions was reduced. Finally, the T-O value was also decreased to 6,645 and 689. From the results described above, we can see that each of the behavior and congestion potentials did not negate the effect of the other one. Therefore, the effectiveness of the motion planning technique using the repulsive potential for the human safety and robot efficiency was shown.

6 Human Sensing System

6.1 Behavior Potential Based on Measurement

In order to apply the behavior potential to humans in a real environment, the robot system is required to measure the actual human behavior. For this purpose, we developed a human sensing system with Microsoft Kinect as shown in Fig. 6. The system, thus, enables to measure the human position, velocity, and direction.

A parameter, κ, used in Eq. (2) is a measure of concentration on the moving direction of humans. This is derived as follows: $\kappa = SD \times ||\boldsymbol{v}||$. In the previous section, the simulation experiments assumed the constant statistical direction, $SD = 5.0$, for the linearly moving humans. However, this is not a distant assumption because actual human behavior varies among different individuals. In addition, it is not always true that the humans move linearly.

In order to provide the behavior potential taking a variety of the human behavior into account, we focus on the statistical direction of the actual humans, $SD(t)$. From the measured position, the moving direction in each image is obtained. Given the moving direction in the n-th image is μ_n, the following function of $SD(t)$ is defined:

$$A(SD(t)) = \frac{1}{N}\sum_{n=1}^{N}\cos(\mu_n - \mu_0^{ML}), \tag{8}$$

where N is the total number of accumulated data μ, and μ_0^{ML} is a maximum likelihood estimator. μ_0^{ML} is calculated from the following equation based on the moving direction of the humans μ obtained in the n-th image: $\mu_0^{ML} = \tan^{-1}\frac{\sum_{n=1}^{N}\sin\mu_n}{\sum_{n=1}^{N}\cos\mu_n}$.

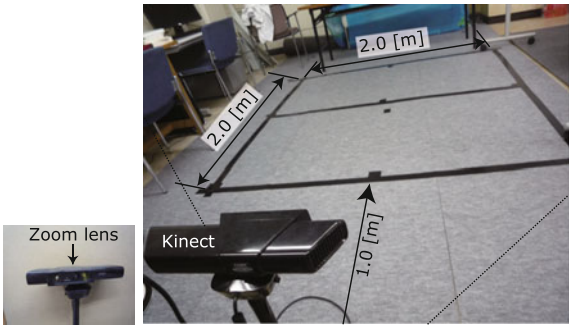


Fig. 6 Developed human sensing system with Microsoft Kinect. A square space with 2×2 [m] was prepared in our laboratory. An optical lens, i.e., Zoom for Kinect, produced by NYKO was used to make a wide view angle. Although the lens reduces the sensing range, the immeasurable area around Kinect is decreased to 1.0 [m]

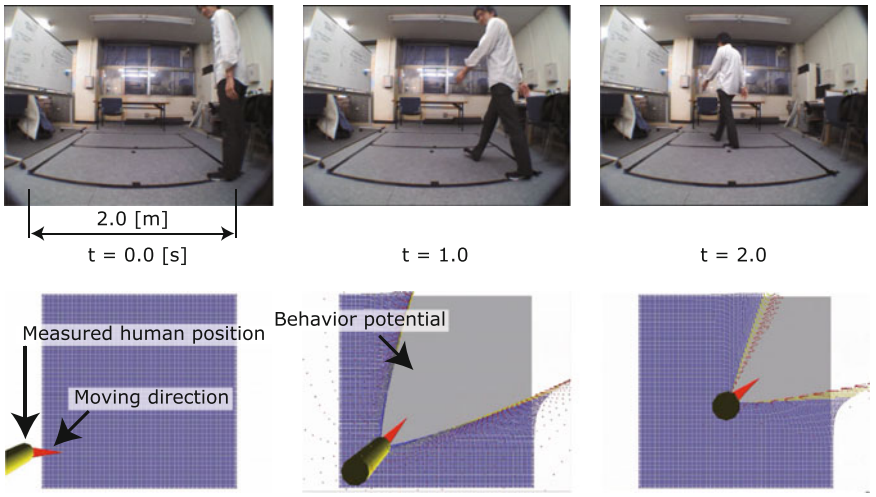


Fig. 7 Original images (*upper*) and measured human position projected to simulation space (*lower*)

Therefore, we can statistically calculate $A(\cdot)$ from the measured human data. On the other hand, $A(\cdot)$ is also defined by the following non-linear function:

$$A(SD(t)) = \frac{I_1(SD(t))}{I_0(SD(t))}, \tag{9}$$

where $I_1(SD(t))$ represents the modified Bessel function of the first kind.

From Eqs. (8) and (9), $SD(t)$, which is an inverse function of $A(\cdot)$, is found by using the Newton-Raphson method. Finally, the behavior potential based on the actual human, $\hat{\kappa} = SD(t) \times ||\boldsymbol{v}(t)||$, is provided as shown in Fig. 7.

Kinect measures the human position in the original images. From the position data in each image and the frame rate, moving velocity and direction of the human are obtained. Simultaneously, the position, velocity, and direction data are sent to the developed simulator, and the behavior potential based on the measurement is generated in the simulation space.

6.2 *Potential Generated from Actual Human Behavior*

To discuss the significance of taking the actual human behavior into account, two behavior potentials U_{o_h} based on κ and $\hat{\kappa}$ were provided for humans in this experiment. For the robots, the congestion potential U_{o_r} was provided. As the actual human data, the following three behavior patterns were measured beforehand: curve, zigzag, and straight as shown in Fig. 8. Figure 9 shows the two behavior potentials for the zigzag behavior.

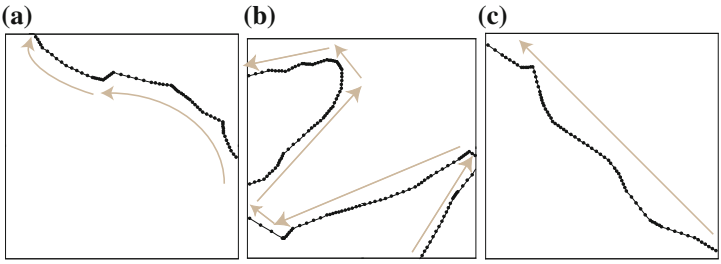


Fig. 8 Measured trajectories of the actual human depending on the behavior patterns. **a** Curve. **b** Zigzag. **c** Straight

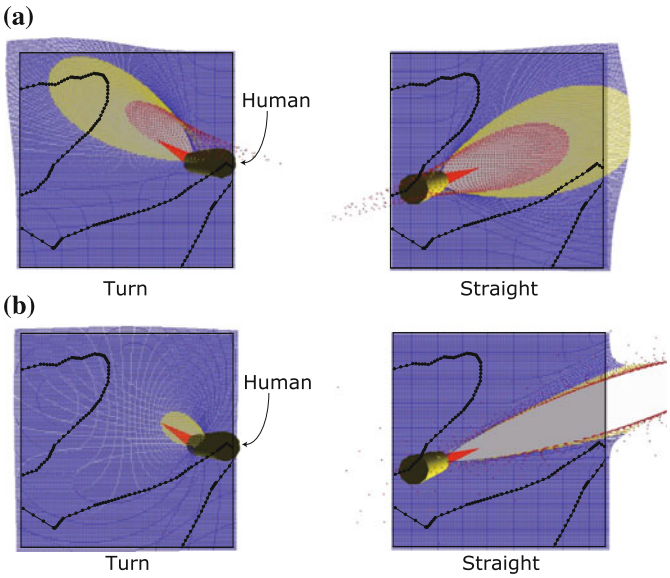


Fig. 9 Behavior potential U_{o_h} changed in accordance with zigzag behavior shown in Fig. 8b. **a** U_{o_h} based on κ. **b** U_{o_h} based on $\hat{\kappa}$

In Fig. 9a, the behavior potential was generated in the moving direction depending on the velocity of the human. However, the shape of the potential was almost constant regardless of the turn and straight behaviors. On the other hand, in Fig. 9b, since the statistical direction, $SD(t)$, was decreased when the human made a turn to the left, the lower shape of the potential was provided over a wide range around the human. In contrast, when the human moved in a straight direction, the potential with an elongate shape was provided to the human.

A behavior pattern from the three shown in Fig. 8 was selected in a random manner, and the human based on the behavior pattern was used repeatedly. Although only one human was used in the simulation space with 2×2 [m], two and six robots moved between their destinations as illustrated in Fig. 5a, b. The total number of

Table 4 One human and two robots

	Time [s]	Collisions	Time×Collisions	SD_R
κ	248.3	21.0	5,214	10.2
$\hat{\kappa}$	287.9	4.6	1,324	6.90

Table 5 One human and six robots

	Time [s]	Collisions	Time×Collisions	SD_R
κ	192.9	4.8	926	9.6
$\hat{\kappa}$	232.9	2.7	629	8.6

accumulated data in Eq. (8) was $N = 20$. Other experimental conditions were as described in Sect. 5.1.

Tables 4 and 5 show the averaged results of five simulations, which are the time, collisions, and T-O value. In addition, the mean value of the statistical direction of all the robots, SD_R is listed. The larger SD_R indicates that a directional change of the robots during the motion planning is smaller. Therefore, robots with the larger SD_R move toward their destinations more linearly than those with the smaller SD_R.

As well as the results listed in Tables 1, 2 and 3, it is noticeable that the number of collisions was not increased even if the number of robots was increased. This is because that the time with the six robots was shorter than that with the two robots.

In both the tables, the behavior potential based on $\hat{\kappa}$ resulted in the longer time for the actual human behavior. This is because that the robots in the potential field based on κ performed the collision avoidance motions with fewer changes in direction. As a result, the statistical directions of the robots were higher than those of the robots in the potential field based on $\hat{\kappa}$. In this regard, however, $\hat{\kappa}$ successfully reduced the number of collisions. Finally, the T-O values were also decreased.

From the results described above, the developed sensing system was shown to be sufficient to provide the behavior potential for the actual humans in a real environment. Moreover, the significance of the potential based on the actual human behavior $\hat{\kappa}$ was shown.

7 Conclusions

This paper focused on systems in which multiple mobile robots exist together with humans. For such systems, in consideration of human safety and robot efficiency, a novel motion planning technique was proposed. For generating repulsive force on robots, two artificial potentials were presented. Through simulation experiments, the effectiveness of the behavior and congestion potentials used in the motion planning technique for the human safety and robot efficiency was shown. Moreover, a sensing

system for humans in a real environment was developed. Finally, the significance of the potential generated from the actual human behavior was shown.

References

1. Yamada, Y.: Safety robot technology in the future. Adv. Robot. **23**(11), 1513–1516 (2009)
2. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
3. Kim, D.H., Shin, S.: New repulsive potential functions with angle distributions for local path planning. Adv. Robot. **20**(1), 25–47 (2006)
4. Ge, S.S., Cui, Y.J.: Dynamic motion planning for mobile robots using potential field method. Auton. Robots **13**(3), 207–222 (2002)
5. Huang, L.: Velocity planning for a mobile robot to track a moving target—a potential field approach. Robot. Auton. Syst. **57**(1), 55–63 (2009)
6. Hoshino, S., Seki, H.: Multi-robot coordination for jams in congested systems. Robot. Auton. Syst. **61**(8), 808–820 (2013)
7. Maniccam, S.: Adaptive decentralized congestion avoidance in two-dimensional traffic. Physica A **363**(2), 512–526 (2006)
8. Burstedde, C., et al.: Simulation of pedestrian dynamics using a two-dimensional cellular automaton. Physica A: Stat. Mech. Appl. **295**(3–4), 507–525 (2001)
9. Pimenta, L.C.A., et al.: Control of swarms based on hydrodynamic models. In: IEEE International Conference on Robotics and Automation, pp. 1948–1953 (2008)
10. Okada, M., et al.: Human Swarm modeling in exhibition space and space design. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 25–30 (2011)
11. Reif, J.H., Wang, H.: Social potential fields: a distributed behavioral control for autonomous robots. Robot. Auton. Syst. **27**(3), 171–194 (1999)

DisCoF: Cooperative Pathfinding in Distributed Systems with Limited Sensing and Communication Range

Yu Zhang, Kangjin Kim and Georgios Fainekos

Abstract Cooperative pathfinding is often addressed in one of two ways in the literature. In fully coupled approaches, robots are considered together and the plans for all robots are constructed simultaneously. In decoupled approaches, the plans are constructed only for a subset of robots at a time. While decoupled approaches can be much faster than fully coupled approaches, they are often suboptimal and incomplete. Although there exist a few decoupled approaches that achieve completeness, global information (which makes global coordination possible) is assumed. Global information may not be accessible in distributed robotic systems. In this paper, we provide a window-based approach to cooperative pathfinding with limited sensing and communication range in distributed systems (called DisCoF). In DisCoF, robots are assumed to be fully decoupled initially, and may gradually increase the level of coupling in an online and distributed fashion. In some cases, e.g., when global information is needed to solve the problem instance, DisCoF would eventually couple all robots together. DisCoF represents an inherently online approach since robots may only be aware of a subset of robots in the environment at any given point of time. Hence, they do not have enough information to determine non-conflicting plans with all the other robots. Completeness analysis of DisCoF is provided.

Keywords Distributed robot systems · Cooperative pathfinding

Y. Zhang (✉) · K. Kim (✉) · G. Fainekos (✉)
School of Computing, Informatics and Decision Systems Engineering,
Arizona State University, Tempe, USA
e-mail: yzhan442@asu.edu

K. Kim
e-mail: Kangjin.Kim@asu.edu

G. Fainekos
e-mail: fainekos@asu.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_23

1 Introduction

Cooperative pathfinding for multi-robot systems has many applications. However, this problem is fundamentally hard in general (i.e., PSPACE-hard [7]). Previous approaches often address this problem in one of two ways. In fully coupled approaches, all robots are considered together and the plans for them are constructed simultaneously. However, given that the complexity grows exponentially with the number of robots, these approaches can easily become impractical. As a result, recent research more often concentrates on decoupled approaches. In decoupled approaches, the plans are constructed (partially or fully) only for a subset of robots at a time; the remaining robots must then take the others' constructed plans into account when constructing their own plans. While decoupled approaches are often suboptimal and incomplete, they typically run much faster than fully coupled approaches, since the number of robots that need to be coupled can be significantly smaller. While there are decoupled approaches that achieve optimality and completeness, they all assume global information, which implies global coordination. However, global information may not be accessible in distributed robotic systems, since such systems are often subject to limited sensing and communication range. As a result, these approaches cannot be implemented on many distributed systems.

In this paper, we introduce a window-based approach for cooperative pathfinding in distributed systems, called DisCoF, with the window size corresponding to the limited sensing and communication range. DisCoF is inherently online, since a robot may not be aware of all the other robots in the environment at any given point of time, let alone determining a non-conflicting plan with them. In DisCoF, a robot can only communicate directly with robots within its sensing range (i.e., *local window*) to coordinate. However, two robots can communicate indirectly through other robots using a message relay protocol. All robots are assumed to be fully decoupled initially: they plan and execute independently and simultaneously. Robots can gradually increase the level of coupling in an online and distributed fashion.

To reduce computation, we need to determine when to couple robots and only couple them when necessary. We follow an intuitive approach to achieving this: couple robots only when they have potential conflicts (i.e., *predictable conflicts* in DisCoF). Furthermore, to efficiently reduce the possibility of future coupling given only local knowledge, instead of making full plans to final goals, robots in each coupling only plan to *local goals* that minimize conflicts within a pre-specified horizon. This process also ensures that these robots make progress to final goals.

However, given the localized nature of this approach, it is subject to live-locks. We identify a live-lock when robots in a coupling cannot make further progress to final goals within the finite horizon, which is a necessary (but insufficient) condition to detect live-locks. Note that detecting live-locks requires global information in general. DisCoF allows "live-locks" to be detected and resolved in a distributed manner. When a live-lock is detected, robots in the coupling use a technique, called *Push and Pull*, to keep within each other's sensing and communication range, while progressing to final goals one at a time.

By combining these methods, DisCoF achieves an efficient solution that also guarantees completeness. Note that for a problem instance that requires global information, DisCoF solves it by eventually coupling all robots together. To the best of our knowledge, this is the first work that guarantees completeness for cooperative pathfinding in distributed systems with limited sensing and communication range. The remainder of this paper is organized as follows. After a brief review of related literature in Sect. 2, we introduce DisCoF in Sect. 3. The live-lock resolution technique is discussed separately in Sect. 4. Conclusions and discussions of future work are presented afterwards.

2 Related Work

The most convenient way to address cooperative pathfinding is to consider robots as fully coupled, since then many existing state-space search algorithms (e.g., A*) can be applied. While this fully coupled search is intractable, approaches have been provided to reduce the branching factors to improve the performance, e.g., [16]. There are also approaches that compile cooperative pathfinding problems into related problem formulations [1, 5, 9, 20] (e.g., maximum flow [20]), and then apply the corresponding algorithms to solve them. However, these approaches are unscalable due to the large state space. Methods for spatial abstraction to reduce the state space have also been discussed [14, 18], but they often suffer optimality and even completeness. By restricting the underlying graphs of problem instances to have certain topologies, optimal solutions can be found fast [12, 13, 19].

More research has been concentrated on decoupled approaches due to its better scalability. One commonly used approach is the hierarchical cooperative A* (HCA* [15]), which is a prioritized planning method. HCA* chooses fixed priorities for robots and makes a plan for a single robot at a time based on its priority, while respecting the computed plans for robots of higher priorities. This process is performed through the use of a reservation table that all robots can access. To reduce the influence of the computed plans for robots of higher priorities (on robots of lower priorities), a windowed HCA* approach (WHCA*) is also discussed in [15]. In WHCA*, robots only send the portions of their plans within a fixed window size (from their current locations) to the reservation table, which has been shown to enable WHCA* to solve more problem instances. More recently, an extension of WHCA* (CO-WHCA* [2]) is proposed, which improves over WHCA* by only reserving plans when there are conflicts. Another common decoupled approach is to create traffic laws for the robots to follow, e.g., [8], thus reducing the possibility of conflicts. Although these decoupled approaches can often find solutions fast, they are incomplete.

There are decoupled approaches that achieve completeness and optimality, e.g., [16, 17]. However, these approaches are still intractable for many problem instances due to the inherent complexity. Hence, more recent approaches often relax optimality while maintaining completeness [4, 10]. In [10], a Push and a Swap operation are

introduced, which are used to move robots to their goals one at a time; the resulting individual plans are then optimized for all robots. The authors in [4] further introduce a new operation, Rotate, to complement Push and Swap, in order to guarantee completeness in more general problem instances. These approaches, however, assume global information (e.g., the individual plans of all robots at any time). Global information may not be accessible in distributed robotic systems with sensing and communication range, in which each robot must create its individual plan based on its local knowledge (including the sensed and communicated information).

In this paper, we introduce an approach that achieves completeness without assuming global information in distributed systems. While cooperative pathfinding with limited sensing and communication range has been investigated before, e.g., [3, 11, 21], to the best of our knowledge, guarantee of completeness has never been provided. Note that cooperative pathfinding with only local knowledge can be considered as a special case of pathfinding with dynamic obstacles. The difficulty lies partly in the existence of live-locks, as global information is required to detect live-locks in general. To achieve completeness, our approach allows robots to gradually increase the level of coupling when potential live-locks are detected.

3 DisCoF

3.1 Problem Formulation

Given a graph $G = (V, E)$ and a set of robots $\mathcal{R}$, the initial locations of the robots are denoted as $\mathcal{I} \subseteq V$ and the goals are denoted as $\mathcal{G} \subseteq V$. Edges in E are undirected. Any robot can move to any adjacent vertex in one time step or remain where they are. A plan $\mathcal{P}$ is a set of individual plans of robots, and $\mathcal{P}[i]$ denotes the individual plan for robot $i \in \mathcal{R}$. Each individual plan is composed of a sequence of actions. For simplicity of presentation, each action is identified by the next vertex to be visited. We denote by $\mathcal{P}_k[i]$ ($k \geq 1$) the action to be taken at time step $k-1$ (or the vertex to be visited at k) for robot i, and by $\mathcal{P}_{k,l}[i]$ ($k \leq l$) the subplan that results by considering only the actions $\mathcal{P}_k[i]$ up to $\mathcal{P}_l[i]$. The goal of cooperative pathfinding is to find a plan $\mathcal{P}$ such that robots start in $\mathcal{I}$ and end in $\mathcal{G}$ after executing it, without any conflicts. The set of locations of robots at time step k is denoted by $\mathcal{S}_k$, and the set of locations of robots after executing a plan $\mathcal{P}$ from $\mathcal{S}$ is denoted by $\mathcal{S}(\mathcal{P})$. Thus, we have $\mathcal{S}_0 = \mathcal{I}$, $\mathcal{S}_0(\mathcal{P}) = \mathcal{G}$ and $\mathcal{S}_k = \mathcal{S}_0(\mathcal{P}_{1,k})$. A conflict happens at time step k, if two robots are in the same location, or their locations at $k-1$ are exchanged. Formally,

$$\mathcal{S}_k[i] = \mathcal{S}_k[j] \vee (\mathcal{S}_k[i] = \mathcal{S}_{k-1}[j] \wedge \mathcal{S}_{k-1}[i] = \mathcal{S}_k[j]) \tag{1}$$

in which $i \in \mathcal{R}$, $j \in \mathcal{R}$ and $i \neq j$.

Each robot can independently compute a plan (without considering other robots) to a given goal from a starting location using a shortest-path planner. For simplicity, we assume that when given the same starting location and goal to different robots, the computed shortest-path plans are the same. Hence, we can denote the shortest-path plan that moves a robot from vertex u to v as $P(u, v)$. The length of $P(u, v)$ is denoted as $\mathcal{C}(u, v)$, i.e., $\mathcal{C}(u, v) = |P(u, v)|$. Furthermore, we make the following assumptions:

1. Robots are homogeneous and have the same sensing and communication range (this assumption only simplifies the presentation and it can be relaxed).
2. Robots are equipped with a communication protocol that allows them to efficiently relay messages.
3. Time steps are synchronized (asynchronous time steps are to be investigated in future work).
4. Each robot has full knowledge of the environment, i.e., G.

The individual plans are constructed and updated in an online fashion in DisCoF. Initially, for each robot i, the individual plan is constructed as $\mathcal{P}[i] = P(\mathcal{I}[i], \mathcal{G}[i])$. Robots then start executing their individual plans until conflicts can be predicted (discussed later). In such cases, the individual plans of robots that are involved are updated from $\mathcal{P}_{k+1}$ to avoid these conflicts, given that the current time step is k.

3.2 Local Window

While the window size in WHCA* [15] is a parameter to determine the number of next plan steps to be sent by each robot to the reservation table, the window size in DisCoF represents the sensing range of the robot. To reduce communication, we only allow a robot to directly communicate with other robots that it can see. However, two robots can communicate indirectly through other robots to coordinate using the message relay protocol. This window is called a *local window* in DisCoF, which is used in the prediction and resolution of potential conflicts.

Definition 1 (*Local Window*) At time step k, the local window of robot $i \in \mathcal{R}$, denoted by $\mathcal{W}_k[i]$, is defined as $\mathcal{W}_k[i] = \{v \in V \text{ (vertices in } G) \mid v \text{ can be reached by } i \text{ from its current location (i.e., } \mathcal{S}_k[i]) \text{ in } \lambda \text{ steps}\}$, in which λ is the window size, a positive integer that is greater than 1.

When a robot j satisfies $\mathcal{S}_k[j] \in \mathcal{W}_k[i]$, we write $i \rhd_k j$ to indicate that robot i can communicate with robot j. A simplifying assumption made here is that the visibility of the sensor is only influenced by the distance, which can be relaxed. Given our assumptions, $\rhd_k$ is symmetric, i.e., i and j can communicate with each other. We indicate this symmetric relation as $i \lhd\rhd_k j$. Furthermore, given the communication relay protocol, $\lhd\rhd_k$ also defines a transitive relation. Namely, if $i \lhd\rhd_k r$ and $r \lhd\rhd_k j$, we also have that $i \lhd\rhd_k j$. The $\lhd\rhd_k$ relation introduces the *coordination graph*.

Definition 2 (*Coordination Graph*) At time step k, the coordination graph $G_k^* = (V_k^*, E_k^*)$ of the robots is constructed as follows:

- $V_k^* = \mathcal{R}$.
- $(i, j) \in E_k^*$ if and only if $i \lhd\rhd_k j$.

Note that the coordination graph is only a structure introduced to facilitate our following discussions. In DisCoF, robots are not required to compute this graph at any time step. Next, we partition the coordination graph into disconnected components that indicate which robots communicate with each other.

Definition 3 (*Outer Closure (OC)*) At time step k, the coordination graph G_k^* is partitioned into disjoint connected subgraphs. Denote Φ_k as the set of vertex sets of these subgraphs. Then, for any $(\phi^x, \phi^y) \in \Phi_k \times \Phi_k$, the following is satisfied:

$\forall (i, j) \in \mathcal{R} \times \mathcal{R}$, if $i \neq j \wedge i \in \phi^x \wedge j \in \phi^y$ holds, we have $i \lhd\rhd_k j$, if and only if $x = y$. Each $\phi \in \Phi_k$ defines an *outer closure*.

Since two robots in different outer closures (OCs) do not communicate in DisCoF (whether directly or indirectly), they do not know about each other's current plan or location (they may not even be aware of each other). Hence, only robots within the same OCs can coordinate with each other.

Definition 4 (*Predictable Conflicts*) At time step k, given an OC $\phi \in \Phi_k$, we define that a robot $i \in \phi$ has a *predictable conflict* with parameter δ, if it would be involved in a conflict at $k + \delta$ ($\delta \leq \beta$, in which β is a pre-specified finite horizon) with another robot in ϕ, and that i would not be involved in any conflicts with robots in ϕ at any time step earlier than $k + \delta$, assuming that robots in ϕ continue with their current individual plans.

The reason for imposing the finite horizon β is due in part to the limited sensing range (i.e., visibility) of the robots since resolution for potential conflicts in the far future is likely to only waste computation resource and time. Note that λ (i.e., window size) and β do not have to be related.

At time step k, if a robot i has a predictable conflict with parameter δ (see Definition 4), we denote it as $\Delta_k^i(\delta)$. We also use Δ_k^i when the parameter does not need to be identified. Predictable conflicts are associated with the notion of inner closure.

Definition 5 (*Inner Closure (IC)*) At time step k, the IC ψ of a given OC $\phi \in \Phi_k$ is the set of robots that satisfy: $\psi = \{i \mid \Delta_k^i \wedge i \in \phi\}$.

Similarly, we denote Ψ_k as the set of ICs for the OCs (there is a one-to-one correspondence) at time step k. Note that the IC of an OC may be empty. We provide an example of OC and IC below.

Example 1 Figure 1 visualizes such a scenario for $\lambda = 2$ and $\beta = 2$. Robots are shown at their initial locations at time step 0. The arrows indicate their respective individual plans for the next few steps and the highlighted gray areas their local windows. We have $r_2 \lhd\rhd_0 r_3$ and $r_3 \lhd\rhd_0 r_4$ and hence $r_2 \lhd\rhd_0 r_4$ through r_3. Thus,

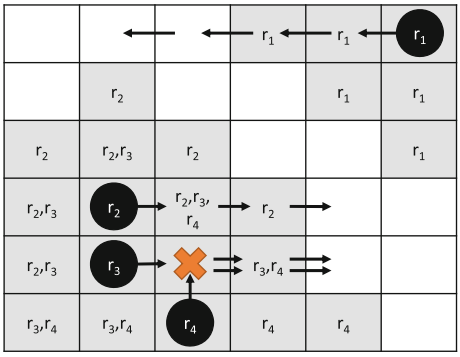


Fig. 1 Scenario that illustrates OC and IC. Two OCs are present, and one of them contains a predictable conflict. Here, r_i indicates the sensing and communication range of each robot i

the OC at time 0 are $\phi^1 = \{r_2, r_3, r_4\}$, and $\phi^2 = \{r_1\}$. Even though r_2, r_3, r_4 are in the same OC, only r_3 and r_4 belong to the corresponding IC, since there is a predictable conflict with parameter 1. Hence, $\psi^1 = \{r_3, r_4\}$ and $\psi^2 = \emptyset$.

3.3 Coupling in OC

Given an OC with predictable conflicts, the goal of coupling is to update the individual plans of robots to proactively resolve these conflicts while avoiding introducing new conflicts in the finite horizon (i.e., specified by β).

At time step k, suppose that conflicts are predicted in an OC $\phi \in \Phi_k$, robots in ϕ need to update their individual plans from $\mathcal{P}_{k+1}$. Note that robots may join and leave different couplings during the online planning process. To make sure that robots make progress to their final goals as a team, we associate a *contribution value* γ with each robot, which captures the individual contribution of the robot to updating the summation of (shortest) distances between all robots' current locations and their final goals. Initially, this value is zero (i.e., $\forall i, \gamma_0[i] = 0$). For robot i, we denote this value just before the coupling at k by $\gamma_{k-}[i]$ and after δ-steps by $\gamma_{k+\delta}[i]$. When δ is 0, γ represents the updated value immediately after the coupling at k (see below).

First, robots in ϕ compute a plan $\mathcal{Q}$ (such that $|\mathcal{Q}| \leq \beta$) which satisfies the following two conditions:

$$\sum_{i \in \phi} \mathcal{C}(\mathcal{S}_k[i], \mathcal{G}[i]) + \gamma_{k-}[i] > \sum_{i \in \phi} \mathcal{C}(\mathcal{S}_k[i](\mathcal{Q}[i]), \mathcal{G}[i]) \tag{2}$$

$$\forall i \in \phi, \neg \Delta_k^i \tag{3}$$

in which Δ_k^i is computed based on the updated individual plans that are constructed as follows: the new individual plan $\mathcal{P}[i]$ for robot $i \in \phi$ is constructed by replacing actions starting from $\mathcal{P}_{k+1}[i]$ by $\mathcal{Q}[i] + P(\mathcal{S}_k[i](\mathcal{Q}[i]), \mathcal{G}[i])$. Here, $\mathcal{S}_k[i](\mathcal{Q}[i])$ is

the *local goal* for i, which is the location of i (currently at $\mathcal{S}_k[i]$) after executing $\mathcal{Q}[i]$. The $+$ symbol is used here to denote concatenation.

At time step k, and after executing each action in $\mathcal{Q}[i]$, the contribution value for i is updated as follows, until a conflict is predicted or this value becomes 0:

$$\gamma_{k+\delta}[i] = \mathcal{C}(\mathcal{S}_k[i](\mathcal{Q}[i]), \mathcal{G}[i]) - \mathcal{C}(\mathcal{S}_{k+\delta}[i], \mathcal{G}[i]) \tag{4}$$

in which $0 \leq \delta \leq |\mathcal{Q}|$, is the steps after the coupling (i.e., number of actions in $\mathcal{Q}$ that are executed). Note that $\mathcal{S}_{k+\delta}[i] = \mathcal{S}_0[i](\mathcal{P}_{1,k+\delta}[i])$ in Eq. (4) is the location of robot i at time step $k+\delta$ under the updated individual plan $\mathcal{P}[i]$ at time step k.

Lemma 1 *Planning (i.e., the computation of $\mathcal{Q}$) in the coupling process converges $\mathcal{R}$ to their final goals as k grows, if Eq. (2) can always be satisfied.*

Proof From Eqs. (2) and (4), we have the following holds:

$$\sum_{i \in \phi} \mathcal{C}(\mathcal{S}_k[i], \mathcal{G}[i]) + \gamma_{k-}[i] > \sum_{i \in \phi} \mathcal{C}(\mathcal{S}_{k+\delta}[i], \mathcal{G}[i]) + \gamma_{k+\delta}[i] \tag{5}$$

First, Eq. (5) holds for all robots that are still executing the coupled plan (i.e., $\mathcal{Q}$) to move to their local goals; furthermore, Eq. (5) also holds for robots that have already reached their local goals or robots that have not engaged in any coupling yet. As a result, Eq. (5) holds for $\mathcal{R}$. As k grows, we know that $\sum_{i \in \mathcal{R}} \mathcal{C}(\mathcal{S}_{k+\delta}[i], \mathcal{G}[i]) + \gamma_{k+\delta}[i]$ would gradually decrease. This also means that $\sum_{i \in \mathcal{R}} \mathcal{C}(\mathcal{S}_k[i](\mathcal{Q}[i]), \mathcal{G}[i])$ would gradually decrease from Eqs. (2) and (4). Given $|\mathcal{Q}| \leq \beta$, the conclusion holds.

Assuming that the condition in Eq. (2) holds, Lemma 1 shows that planning converges to final goals for $\mathcal{R}$. This condition requires robots in a coupling to always make progress jointly within the finite horizon. However, this assumption does not hold in the presence of live-locks. In such cases, robots in a coupling would eventually be unable to find a $\mathcal{Q}$ that satisfies both Eqs. (2) and (3).[1] Furthermore, the limited horizon can also cause the search of $\mathcal{Q}$ to fail. However, we realize that when live-locks are present in distributed systems with only local knowledge, the search is bound to fail eventually even with unlimited horizon. Hence, we do not distinguish the two causes, and consider it as a "live-lock" being detected when when Eq. (2) becomes unsatisfiable (while satisfying Eq. (3)).

3.4 Computing $\mathcal{Q}$

Before discussing how "live-locks" are addressed in DisCoF, we provide details on how $\mathcal{Q}$ is computed. Given that coupled search is expensive, we aim to minimize $|\mathcal{Q}|$ as well as the number of robots that need to be coupled.

[1] Note that Eq. (3) can always be satisfied by forcing all the robots in a coupling to stay, which may cause deadlocks. Eq. (2) prevents deadlocks.

To achieve this, we try to construct $\mathcal{Q}$ that satisfies Eqs. (2) and (3) for ρ ($\rho \subseteq \phi$), which is initially set to be the corresponding IC for ϕ, while forcing robots in ρ to respect the plans (i.e., avoiding predictable conflicts) of robots in $\phi \setminus \rho$ in the next β steps. Note that having ρ instead of ϕ satisfy Eq. (2) does not influence the planning convergence.

The search first checks $\mathcal{Q}$ for ρ with $\theta = 1$, in which $\theta = |\mathcal{Q}|$, and gradually increases θ until $\theta = \beta$. If a valid $\mathcal{Q}$ is found for the current θ, the $\mathcal{Q}$ is returned. Otherwise, if $\phi \setminus \rho \neq \emptyset$, ρ is expanded to include robots in $\phi \setminus \rho$ that are also within the combined region of local windows of robots in ρ, and the current θ is re-checked; else, θ is incremented or unsatisfiability is returned when $\theta = \beta$.

4 Push and Pull

In DisCoF, when unsatisfiability is returned in computing $\mathcal{Q}$ for an OC ϕ, we consider it as a "live-lock" (i.e., robots in ϕ may have contributed in creating a live-lock situation) being detected. To resolve it, information of all robots in ϕ must be accessible. In distributed systems, this requires the robots in ϕ to maintain within each other's sensing and communication range (thus remain coupled). Furthermore, note that a live-lock may not involve all robots in $\mathcal{R}$ and there may be multiple live-locks in the environment. When a live-lock is detected, robots in ϕ form a *coupling group*, ω, which executes a live-lock resolution process described next. This process also allows a coupling group to merge with other groups and robots, thus gradually increasing the level of coupling. In some cases, e.g., when a global live-lock is present, robots in DisCoF can eventually become fully coupled.

4.1 Overview

To achieve completeness, DisCoF uses a technique that is similar to Push and Rotate [4], which we call Push and Pull. To ensure completeness in Push and Rotate, robots must move to goals one at a time according to the priorities of subproblems to which they belong. Robots that have already reached their goals are respected (i.e., considered as obstacles) by the subsequent Push operations. When Push fails, Push and Rotate uses a Swap operation to ensure that these robots move back in their goals as the remaining robots move. Such a priority ordering must also be respected in DisCoF. At time step k, for all coupling groups that have been formed, the basic idea is to: (1) maintain robots in these groups within each other's sensing and communication range; (2) for each group, move robots to goals one at a time based on a relaxed version of the priority ordering, which is consistent to that in Push and Rotate; (3) add robots that introduce predictable conflicts with a coupling group as robots in the group move to their goals. Each coupling group progresses independently of other

robots and coupling groups unless there are predictable conflicts. The main process is described in Algorithm 1.

Algorithm 1 Live-lock Resolution Process in DisCoF for a Coupling Group ω

1: *Current time step is k.*
2: **while** $\exists i \in \omega, \mathcal{S}_k[i] \neq \mathcal{G}[i]$ **do**
3: **if** predictable conflicts detected with other robots **then**
4: Add other robots with predictable conflicts to, or merge their groups with ω.
5: Recompute the priorities of subproblems.
6: **end if**
7: **if** r is not defined $\vee$ robots with a higher priority than r is found $\vee$ r reaches $\mathcal{G}[r]$ **then**
8: $r \leftarrow$ the robot with (equal) highest priority in ω.
9: **end if**
10: Push and Pull r to $\mathcal{G}[r]$.
11: **end while**

In Algorithm 1, the coupling of robots in ω is maintained by the Push and Pull technique. As a result, predictable conflicts can only be introduced by other robots. When a coupling group detects predictable conflicts with another group, two groups are merged. Furthermore, when a robot that has already reached its goal is added to a coupling group in the live-lock resolution process, if the robot's priority is not the highest among all robots that have not reached their goals after recomputing the priorities, this robot is not considered as having reached its goal in Push and Pull. This means that the Push and Pull operations can move these robots. Also, the priority ordering (i.e., the $\prec$ relations in [4]) is maintained and aggregated by the robots whenever new relations are identified (in Line 5); given that the relaxed priority ordering is consistent with that in Push and Rotate, robots can gradually achieve a consensus of this ordering.

4.2 Assigning Priorities

To ensure completeness, the priorities of subproblems in Push and Rotate [4] must be respected. However, given the limited visibility of the robots, this priority ordering can only be partially computed for each coupling group. This partially computed ordering in DisCoF is kept consistent with the priority ordering in Push and Rotate.

To compute the priorities, first, Push and Rotate identifies the subproblems. Since this computation is only dependent on the graph structure, robots in DisCoF can individually identify the set of subproblems.

Next, Push and Rotate assigns robots to subproblems. DisCoF computes a relaxed version of this assignment to ensure that assignments are only made when they are consistent with those in Push and Rotate. Denote the set of subproblems as $\mathcal{D}$. Algorithm 2 presents the algorithm to compute the assignment in a coupling group ω.

Algorithm 2 Algorithm for Assigning Robots to Subproblems in ω

1: **for all** $\mathcal{D}_h \in \mathcal{D}$ **do**
2: **for all** $v \in \mathcal{D}_h$ **do**
3: **for all** $u \notin \mathcal{D}_h$ for which $(u, v) \in E^\omega$ **do**
4: $m' \leftarrow$ number of unoccupied vertices reachable from v in $G^\omega \setminus \{u\}$.
5: $m'' \leftarrow$ number of unoccupied vertices reachable from $\mathcal{D}_h$ in $G^\omega \setminus \{v\}$; $\neg m \leftarrow$ number of unoccupied vertices unreachable from v in $G \setminus \{u\}$, given only robots in G^ω.
6: **if** $(m' \geq 1 \wedge \neg m \geq 1) \vee m'' \geq 1$ **then**
7: Assign robot in v to $\mathcal{D}_h$.
8: **end if**
9: Follow path from u away from v and assign the first $m' - 1$ (all if less than $m' - 1$) on this path to $\mathcal{D}_h$ in G^ω.
10: **end for**
11: **end for**
12: **end for**

The differences of Algorithm 2 from that in Push and Rotate lie in Line 4, 5, 6 and 9. While the computation for these lines is performed based on the global graph (i.e., G) in Push and Rotate, the computation in DisCoF is based on $G^\omega = (V^\omega, E^\omega)$ (which represents the combined region of the local windows of robots in ω), and G given only robots in G^ω. Note that not every robot may be assigned to a subproblem and the unassigned robots are assumed to have the lowest priorities.

Lemma 2 *The assignment of robots to subproblems in DisCoF is consistent to that in Push and Rotate [4]: if a robot r is assigned to subproblem $\mathcal{D}_h$ in Algorithm 2, it is also assigned to $\mathcal{D}_h$ in Push and Rotate.*

Proof We only need to prove that: (1) when the condition in Line 6 is satisfied, the corresponding condition in Push and Rotate is also satisfied; (2) m' and m'' in Algorithm 2 are smaller than those in [4], and $\neg m \geq 1$ in Algorithm 2 implies $m' < m$ in [4]. These directly follow from how they are computed.

In the third step, Push and Rotate assigns priorities to the subproblems. Robots within the same subproblems receive the same priorities. Similarly, the reference of global graph is changed to G^ω; otherwise, the process is unchanged.

Lemma 3 *The assignment of priorities to subproblems in DisCoF is consistent to that in Push and Rotate [4]: if two subproblems $\mathcal{D}_{h1}$ and $\mathcal{D}_{h2}$ satisfy $\mathcal{D}_{h1} \prec \mathcal{D}_{h2}$, they must also satisfy $\mathcal{D}_{h1} \prec \mathcal{D}_{h2}$ in Push and Rotate.*

Proof This conclusion follows almost directly from Lemma 2 and the process for assigning priorities to subproblems.

Note that this assignment process is executed by each coupling group in DisCoF instead of all robots in Push and Rotate. This means that while the assignments are consistent with that in Push and Rotate, they are computed for different (and disjoint) sets of robots in DisCoF.

4.3 Maintaining and Expanding ω

Robots in a coupling group can use the operations (i.e., Push, Swap and Rotate) in Push and Rotate to move to their goals one at a time (for details, refer to [4]). To maintain robots within ω in sensing and communication range, we introduce a new operation, called Pull. Denote r as the current robot that is being moved to its final goal in ω. As r moves to its goal, it can use any of the Push, Swap and Rotate operations. Every step that r moves as a result of these operations, it also invokes the Pull operation on the other robots in ω.

The Pull operation computes a shortest-path plan p from r to any robot $s \in \omega \setminus r$. A set $\mathcal{U}$ is created, which contains only r initially. If p does not pass through other robots in ω, and the first step in p leads s closer to r, s is added to $\mathcal{U}$. If the first step in p does not introduce conflicts with other robots in ω, this step is added to the individual plan of s; otherwise, an action to stay is added to the individual plan of s. For robots that have been newly added into $\mathcal{U}$, they recursively apply the Pull operation on robots that are not in $\mathcal{U}$. This process ends until all robots in ω are in $\mathcal{U}$. The Pull operation is presented in Algorithm 3. Figure 2 illustrates the Pull operation in a simple scenario.

Lemma 4 *The Pull operation maintains robots in each coupling group within each other's sensing and communication range.*

Proof The Pull operation, after execution, ensures that any robot $s \in \omega$ is no further away from one of the robots in ω before its execution. Hence, the conclusion holds.

Similar to Push, Pull may fail (Line 13 in Algorithm 3) since it must respect the robots (with equal or higher priorities) that have already reached their goals. In such cases, a similar procedure using Swaps as for the Push operation in [4] can be used; these Swaps can cause robots that are being swapped to recursively invoke Pull.

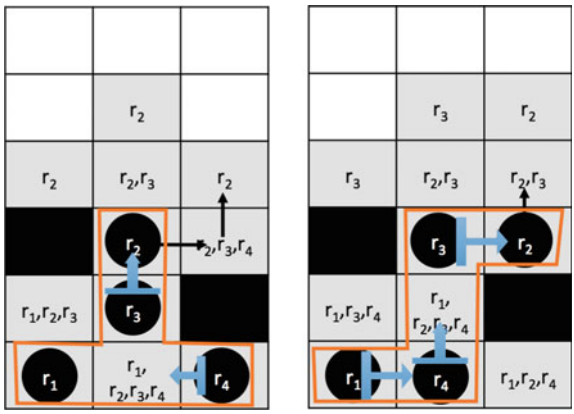


Fig. 2 Scenario that illustrates the Pull operation. Left figure shows that robot r_2 is moving to its goal. Right figure shows the same scenario after one time step. *Blue arrows* show the actions being added to the individual plans of the corresponding robots at each step by the Pull operation

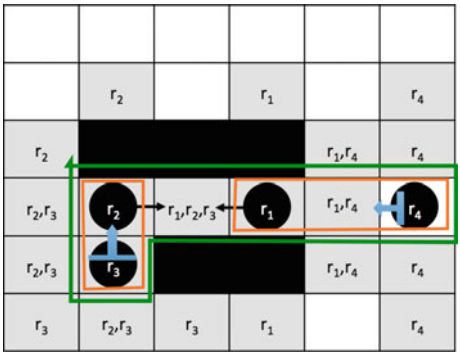


Fig. 3 Scenario that illustrates the expanding process, in which two coupling groups, each with two robots, are merged when a conflict is predicted in the next step

Algorithm 3 Pull operation in ω

1: $\mathcal{U} \leftarrow \{r\}; \mathcal{N} \leftarrow \emptyset$
2: **while** $\mathcal{U} \neq \omega$ **do**
3: **for all** $s \in \omega \setminus \mathcal{U}$ **do**
4: $p \leftarrow P(\mathcal{S}_k[s], \mathcal{S}_k[r])$, consider robots that have reached goals as obstacles.
5: **if** p does not pass through robots in ω that have not reached goals $\wedge$ p moves s closer to r **then**
6: **if** no conflicts with other robots in ω after executing the first step in p **then**
7: Add the first step in p to the individual plan of s.
8: **else**
9: Add an action for s to stay in the next step.
10: **end if**
11: $\mathcal{U} \leftarrow \mathcal{U} \cup \{s\}; \mathcal{N} \leftarrow \mathcal{N} \cup \{s\}$
12: **else**
13: **return** False.
14: **end if**
15: **end for**
16: $r \leftarrow Pop(\mathcal{N})$.
17: **end while**

When there are other robots within the combined region of the local windows of robots in ω, robots must plan to consider predicted conflicts. Each coupling group makes a plan for the next β steps considering only robots in the group. When no conflicts are predicted, robots continue with this plan. When conflicts are predicted, ω is expanded as we previously discussed. The expanded coupling group chooses the robot currently with the (equal) highest priority to move to the goal.[2] Figure 3 illustrates the merge of two coupling groups. In the group on the left (r_2 and r_3), r_2 is moving to its goal, pulling r_3, and in the other group, r_1 is moving to its goal, puling r_4. Since a predictable conflict exists between r_2 and r_1, the two groups are merged.

[2]If more than one robot have the same (highest) priority, we can arbitrarily choose among them.

4.4 Analysis

To prove the completeness of DisCoF, we use a property that is derived directly from Theorem 2 in Push and Rotate [4].

Corollary 1 *If the cooperative pathfinding problem is solvable, the assignment of robots in a coupling group to subproblems remains unchanged unless the group is expanded.*

Theorem 1 *DisCoF is complete for the class of cooperative pathfinding problems in which there are two or more unoccupied vertices in each connected component.*

Proof We provide the proof sketch here, which is based on the following observations: (1) When every coupling group is independent of other robots and groups, DisCoF is complete; this is almost a direct result from Push and Rotate, since the Pull operation does not influence the other operations. (2) When a coupling group is expanded, robots in the group are maintained within each other's sensing and communicating range; this is a direct result from Lemma 4. (3) The priority ordering relations (i.e., $\prec$) are maintained and gradually aggregated (to reach a consensus) as they are identified; this is a result from Lemma 2, Lemma 3 and Corollary 1. (4) Robots with the highest priorities are respected (in Push and Pull operations) by the coupling groups in the Push and Pull process (similar to that in Push and Rotate), which moves robots with the highest priorities to goals first.

Since it has been shown in [4] that robots with the highest priorities must be moved to goals first in order to ensure a solution, these robots must eventually be assigned the highest priorities as the coupling groups move. Hence, these robots would be moved to their final goals. This process then continues to robots with the second highest priorities and so on. Hence, DisCoF is complete.

5 Conclusions

In this paper, we introduce a window-based approach for cooperative pathfinding in distributed systems, with the window size corresponding to the limited sensing and communication range in such systems. This approach, called DisCoF, is an inherently online approach. To limit coupling in order to reduce computation, we introduce a formulation that allows robots to avoid future conflicts while still making joint progress to their final goals. This formulation also allows "live-locks" to be detected; in such cases, we use a Push and Pull technique. We show that DisCoF is complete. To the best of our knowledge, this is the first work that guarantees completeness for cooperative pathfinding with limited sensing and communication range in distributed systems. Note that the general definition of conflict potentially allows DisCoF to be applied to cooperative pathfinding with different robotic platforms, e.g., adding the consideration of height for UAVs.

In future work, we plan to provide a detailed evaluation of DisCoF and compare it with other related approaches. We also plan to extend the formulations to consider more complex environment and goal specifications (e.g., using temporal logic specifications [6]). Other directions include extending the approach to support continuous motions, heterogeneous robots, and asynchronous time steps. For recent progresses, refer to https://cpslab.assembla.com/spaces/discof/.

Acknowledgments This research is supported in part by the ARO grant W911NF-13-1-0023, the ONR grants N00014-13-1-0176 and N00014-13-1-0519, and the NSF award CNS 1116136.

References

1. Ayanian, N., Rus, d., Kumar, V.: Decentralized multirobot control in partially known environments with dynamic task reassignment. In: 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (2012)
2. Bnaya, Z., Felner, A.: Conflict-oriented windowed hierarchical cooperative A*. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (2014)
3. Clark, C.M., Rock, S.M., Latombe, J.-C.: Motion planning for multiple mobile robots using dynamic networks. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 4222–4227, Sep 2003
4. de Wilde, B., ter Mors, A.W., Witteveen, C.: Push and rotate: cooperative multi-agent path planning. In: 12th International Conference on Autonomous Agents and Multiagent Systems (2013)
5. Desaraju, Vishnu R., How, Jonathan P.: Decentralized path planning for multi-agent teams with complex constraints. Auton. Robots **32**(4), 385–403 (2012)
6. Fainekos, Georgios E., Girard, Antoine, Kress-Gazit, Hadas, Pappas, George J.: Temporal logic motion planning for dynamic robots. Automatica **45**(2), 343–352 (2009)
7. Hopcroft, J.E., Schwartz, J.T., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". Int. J. Robot. Res. **3**(4), 76–88 (1984)
8. Jansen, R., Sturtevant, N.: A new approach to cooperative pathfinding. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, pp. 1401–1404, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2008)
9. Liu, L., Shell, A.: Physically routing robots in a multi-robot network: flexibility through a three-dimensional matching graph. Int. J. Robot. Res. **32**(12), 1475–1494 (2013)
10. Luna, R., Bekris, K.: Efficient and complete centralized multirobot path planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2011)
11. Otte, M., Bialkowski, J., Frazzoli, E.: Any-com collision checking: sharing certificates in decentralized multi-robot teams. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (2014)
12. Parker, L.E.: Encyclopedia of Complexity and System Science, Path Planning and Motion Coordination in Multiple Mobile Robot Teams. Springer, New York (2009)
13. Peasgood, M., Clark, C.M., McPhee, J.: A complete and scalable strategy for coordinating multiple robots within roadmaps. IEEE Trans. Robot. **24**(2), 283–292 (2008). April
14. Ryan, M.: Graph decomposition for efficient multi-robot path planning. In Proceedings of the 20th International Joint Conference on Artifical Intelligence, pp. 2003–2008. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007
15. Silver, D.: Cooperative pathfinding. In: Conference on Artificial Intelligence and Interactive Digital Entertainment (2005)

16. Standley, T.: Finding optimal solutions to cooperative pathfinding problems. In: AAAI Conference on Artificial Intelligence (2010)
17. Standley, T., Korf, R.: Complete algorithms for cooperative pathfinding problems. In: Proceedings of the 22nd International Joint Conference on Artifical Intelligence (2011)
18. Sturtevant, N., Buro, M.: Improving collaborative pathfinding using map abstraction. In: Artificial Intelligence and Interactive Digital Entertainment (AIIDE), pp. 80–85 (2006)
19. Wang, K.H.C., Botea, A.: Fast and memory-efficient multi-agent pathfinding. In: International Conference on Automated Planning and Scheduling, pp. 380–387 (2008)
20. Yu, J., LaValle, S.M.: Multi-agent path planning and network flow. In: *Algorithmic Foundations of Robotics X*, vol. 86, pp. 157–173. Springer (2013)
21. Zuluaga, M., Vaughan, R.: Reducing spatial interference in robot teams by local-investment aggression. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005), pp. 2798–2805, Aug 2005

Decentralized Multi-agent Path Selection Using Minimal Information

Andrew Kimmel and Kostas Bekris

Abstract This work studies conflict avoidance between moving, non-communicating agents with minimum sensing information. While safety can be provided by reactive obstacle avoidance methods for holonomic systems, deadlock avoidance requires reasoning over different homotopic paths in cluttered scenes. A method to compute the "interaction cost" of a path is proposed, which considers only the neighboring agents' observed positions. Minimizing the interaction cost in a prototypical challenge with two agents moving through two corridors from opposing sides guarantees the selection of non-conflicting paths. More complex scenes, however, are more challenging. This leads to a study of alternatives for decentralized path selection. Simulations indicate that following a "minimum-conflict" path given the other agents' observed positions provides deadlock avoidance. A scheme that selects between the minimum-conflict path and a set of shortest paths given their interaction cost improves path quality while still achieving deadlock avoidance. Finally, learning to select between the minimum-conflict and one of the shortest paths allows agents to be adaptive to the behavior of their neighbors and can be achieved using regret minimization.

Keywords Multi-agent · Decentralized · Coordination · Path planning

1 Introduction

Advances in robotic technology allow applications where multiple robots operate in the same cluttered environment, potentially also in the presence of people or animals. Explicit communication with the other agents, especially when humans are involved, may not be feasible or desirable. Similarly, it may be difficult to model or predict the actions of the other agents. This motivates decentralized methods that allow a robot

A. Kimmel · K. Bekris (✉)
Rutgers University, Piscataway, NJ, USA
e-mail: andrew.kimmel@cs.rutgers.edu

K. Bekris
e-mail: kostas.bekris@cs.rutgers.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_24

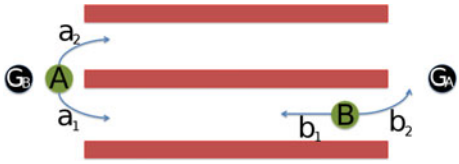


Fig. 1 If both agents A and B insist on following the same corridor, a reactive collision avoidance method may not allow them to make progress to their goals, G_A and G_B

to reach its goal safely given minimal information and without strong assumptions about the intentions of its moving neighbors. Such solutions should avoid deadlocks and minimize executed path length or task completion time.

Challenges, Foundations and Objectives: Avoiding collisions with unexpected obstacles or mobile agents, can be effectively addressed by reactive collision avoidance methods, such as those based on the popular Velocity Obstacle framework [5, 31] or trajectory deformation methods [6, 15]. These methods generally provide smooth, natural-looking paths, but they are primarily local techniques and do not reason about the robot's global path. Although local motion coordination can be achieved [16], if the agents select conflicting paths, reactive collision avoidance can still give rise to deadlocks and poor performance. For instance, consider the situation in Fig. 1, where two robots on opposing sides of two corridors need to exchange positions. If both robots decide to move along the lower corridor, e.g., because it corresponds to their individual shortest path, the space is narrow enough to prevent the robots from crossing.

Robots in such situations that replan [23] and change their path to a different homotopy class [3, 12] can potentially resolve conflicts. Such coordination can be achieved by assuming that the robots share information [1], or follow a form of centralized planning [24], or by respecting a set of pre-specified "social" rules [19], or performing sophisticated prediction [28, 32], agent modeling [25, 26], learning [11] or game-theoretic reasoning [13]. The information required to use such solutions may correspond (a) to the actions selected by neighbors, (b) the utilities of different motions, (c) the goals of neighbors, or (d) extensive prior experience interacting with other agents. Such information is difficult to attain quickly and reliably, especially when a robot interacts with a human, since the robot has little knowledge about the human's future actions without explicit communication. Furthermore, it is interesting to study what is achievable without any prediction, intent recognition or modeling of the moving agents.

Considered Methodology: This work employs strictly decentralized methods while utilizing minimal information. Each robot has access only to the current position and velocity of its neighbors from sensing data. The basic framework assumes that robots replan paths frequently [23] and employ reciprocal velocity obstacles [27, 30] so as to follow these paths while avoiding collisions. The velocity information is actually used only for the adopted reactive obstacle avoidance method based on Velocity Obstacles [31] and not for the proposed path planning techniques. Learning is also considered, corresponding to online learning of appropriate strategies in response

to the behavior of other agents. The considered techniques are evaluated in various simulated benchmarks.

If agents greedily select the globally shortest path to their goal, this frequently results in deadlocks as in the case of Fig. 1. To address this issue, one alternative is to consider a set of diverse paths instead of only the shortest one. The notion of path diversity has been shown to be helpful in many different challenges, but frequently corresponds to a local concept [7, 18]. One way to compute a diverse set of global paths is to consider different homotopy classes using search-based primitives [2, 3] over an underlying roadmap. This work provides a method for selecting a minimally conflicting path out of this set by defining an "interaction cost" for each path given the other agents' current positions. This process is designed so as to address the issue in the prototypical example provided of Fig. 1.

Computing a large number of diverse paths is computationally challenging in complex scenes. If, instead, only a small set of k shortest paths in distinct homotopy classes is considered deadlocks can still arise. An alternative is to compute the "minimum-conflict" homotopy class, which minimizes interactions with other agents. This notion is related to the "minimum constraint displacement" problem, which has recently attracted attention [10]. Here, constraints on the minimum-conflict homotopy class correspond to the observed locations of other agents and can be discovered in a computationally efficient manner using an underlying roadmap. The experimental evaluation shows that when the moving agents follow their "minimum-conflict" paths, they can avoid deadlocks even in complex scenes.

While minimum conflict paths achieve deadlock avoidance experimentally, they can be inefficient, forcing agents to follow long paths to reach their goals. Considering both the minimum-conflict path and a set of k shortest paths in distinct homotopy classes typically results in improved performance in terms of path length. This work considers two approaches for choosing between (a) the conservative alternative of following the "minimum-conflict" path and (b) the greedy choice corresponding to one of the k shortest paths. The first approach is a deterministic strategy that selects the path among the $k + 1$ choices, which minimizes the "interaction cost".

The second approach is based on the notion of regret minimization and corresponds to the Polynomial Weights `PW` algorithm [21, 22]. Regret minimization is a favorable way to reason among unpredictable agents, without knowing their goals, utilities, intents, or beliefs, as in the setup of this work. The application of the `PW` algorithm here accumulates regret for the "minimum-conflict" strategy and the "greedy" choice strategy by observing the choices of the other agents in the same workspace and assigning a loss to each strategy in hindsight. A probability is then assigned for selecting each strategy based on the accumulated losses.

Both the deterministic and the learning-based solution result in deadlock avoidance in the simulated benchmarks considered in this work. The learning approach can also adapt to different behaviors by neighboring agents.

Contribution and Overview of Results: The key observations from this work can be summarized as the following points:

(a) Using "interaction cost" to select a path among multiple choices assists in minimizing the occurrence of deadlocks. In a prototypical benchmark with few homotopy classes and agents, it can guarantee deadlock avoidance.
(b) Computing minimum-conflict paths is experimentally shown to be critical for avoiding deadlocks even in more complex scenes with many homotopy classes and is interesting to further analyze the properties of this strategy.
(c) Considering multiple paths in distinct homotopy classes together with the minimum-conflict path results in improved path quality and execution time.
(d) Regret minimization is a computationally efficient way for robots to react to the behavior of other agents in the scene without explicit communication.

2 Problem Setup

Path deconfliction problems can be defined in general configuration spaces but this discussion will focus on holonomic planar navigation as it provides an easy way to describe the framework and corresponds to the accompanying simulations.

Consider a set of m planar, holonomic agents $\{a_1, \ldots, a_m\}$ that move with bounded velocity $v \in [0, v_{max}]$ in the same workspace $\mathbb{W}$. The configuration space of an agent is $\mathbb{Q} = \mathbb{R}^2$, where $\mathbb{Q}_{free}$ represents the obstacle free subset given static obstacles. Given a configuration $q_i \in \mathbb{Q}$, the expression $a(q_i)$ corresponds to the collision volume of agent a_i in $\mathbb{W}$.

A path $\pi_i = \{q_i | q_i : [0, 1] \rightarrow Q_{free}\}$ for agent a_i corresponds to a continuous curve in Q_{free}. Given a time scaling function $\sigma_i : \mathbb{R}^{\geq 0} \rightarrow [0, 1]$ it is also possible to define the sequence of configuration $\tau_i = \pi_i \circ \sigma_i$ that the agent visits at each point in time.

The problem formulation assumes that each agent a_i wants to reach a desired goal $q_i^{\mathbb{G}} \in \mathbb{Q}$ without conflicts. The objective then is for the agents to select the sequence of configurations $\{\tau_1, \ldots, \tau_m\}$ they will follow in a decentralized manner, such that in finite time T: $\forall\, i \in \{1, \ldots, m\} : \tau_i[T] = q_i^{\mathbb{G}}$. Collisions between agents must be avoided, unless one of the agents has reached its goal, i.e.,

$$a(\tau_i[t]) \cap a(\tau_j[t]) = \emptyset \ \vee \ a(\tau_i[t]) = q_i^{\mathbb{G}} \ \vee \ a(\tau_j[t]) = q_j^{\mathbb{G}}.$$

The above description implies a version of the so called "garage" assumption. When agents reach their goal, they are removed from the workspace and are not considered for collisions when other agents pass through that goal.

Agents are never aware of the goal of any other agent or the path selected by another agent. At any point in time an agent can only observe the positions of other agents as long as their configurations are within a certain sensing radius.

Agents are assumed to have access to a collision avoidance method (e.g., Reciprocal Velocity Obstacles [30] are used in the accompanying experiments), which is used to follow their selected path while still avoiding collisions with other agents. This means that the planned path may not be executed perfectly due to the influence of neighboring agents and the use of the obstacle avoidance method.

Note that the above discussion can be easily extended to include the case where one agent is the planning robot that employs a method for achieving deconfliction while all the other agents are unpredictable dynamic obstacles that ignore the presence of the planning agent. In these situations, the relative velocity of the planning agent and the dynamic obstacles should be such so that the collision avoidance method can always guarantee the safety of the planning agent.

The above setup involving unpredictable neighboring agents motivates a replanning framework for recomputing paths given the latest observed configurations of agents. This replanning approach forms the basis of the overall methodology that is described in the following section.

3 Methods

This section first describes a straightforward method for integrating global path planning and local collision avoidance, which, however, can lead to deadlocks in many setups. Then a sequence of alternative strategies for computing the global path are considered so as to avoid such situations.

Replanning Framework: During execution of an action, a robot can deviate from its corresponding planned path due to the reactive collision avoidance executing a safety control. Naïvely following the original planned path is therefore not sufficient, as the robot will most likely not be able to reach its goal as intended. A replanning framework [9, 23] as illustrated in Fig. 2 is used to address such issues. The framework follows related work, where first a roadmap is precomputed using a sampling-based motion planning method and then integrated with a collision-avoidance method [29]. The sampling-based planner used in this work is PRM* [14]. The path computed for time $t-1$ to t will not be executed perfectly, as shown in Fig. 2; however, the framework updates the predicted state of the robot accordingly. By using such a replanning

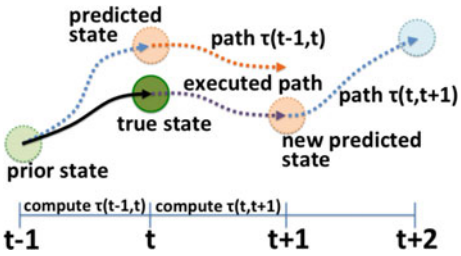


Fig. 2 The path computed $[t-1, t]$ is executed during time $[t, t+1]$. The state at time t can deviate from the predicted initial state for the computed plan, so planning for cycle $[t+1, t+2]$ must start from an updated predicted state

framework, where the agent updates its new predicted state given the perceived differences between its true state and the previous predicted state, the agent becomes robust to perturbations in the solution path caused by the use of reactive methods. The most straightforward path selection process corresponds to selecting the shortest path to the goal ignoring other agents. As argued before, this choice can lead to deadlocks despite the availability of the reactive collision avoidance method, when the shortest paths of two agents conflict.

K-Best Paths from Different Homotopy Classes: Rather than simply selecting the greedy path at each planning cycle, one alternative is for each agent to consider a diverse set of paths and select one that reduces possible interactions with other agents. In an environment with many obstacles and narrow corridors which cause conflicts between agents that cannot be resolved by reactive obstacle avoidance, it makes sense to consider paths that belong to different homotopy classes [3]. Homology classes [4] can also be used to compute a diverse set of paths, however this work does not utilize homologies since they do not differentiate between paths that symmetrically loop around obstacles. When considering 2D problems, paths are in different homotopy classes when the area between them contains an obstacle. A complete definition for homotopy can be found in the related literature [8].

By ignoring paths that loop around obstacles, the set of non-homotopic paths describes all of the shortest-length paths that bring the agent from its current position to the goal. These computations take place over an underlying roadmap, and use a set of search-based primitives. An example of the resulting set of computed paths in a simulated environment is shown in Fig. 3.

Minimizing Interaction Cost The question that arises is how should agents differentiate among the available paths in order to select motions that will allow them to make progress to their goals. To describe the proposed process, consider the situation depicted in Fig. 1, where agent A can follow action a_1 to move through the lower corridor and action a_2 to move through the upper corridor towards its goal G_A. These actions correspond to two solutions returned from the homotopy class computation described in the previous section, regardless of the current configuration of the robot q_A. Similarly, agent B has choices b_1 and b_2.

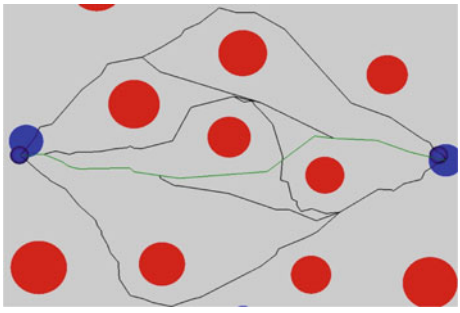


Fig. 3 Selecting the path with the lowest interaction cost from k different homotopy classes is the "k-best" strategy

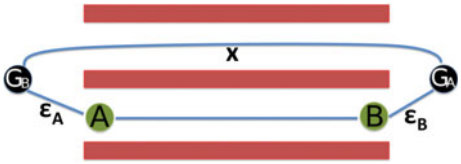


Fig. 4 Backtracking increases path length by ε_A and ε_B for robots A and B respectively, while remaining in the current corridor reduces path length by ε_A and ε_B respectively

Then the question is how costs $C(a_1)$, $C(a_2)$, $C(b_1)$, $C(b_2)$ can be computed appropriately, and in a decentralized manner, so that in any situation the two agents will decide to follow different corridors when they try to select the action with minimum cost. The most conflicted situation occurs when both robots are already following the same corridor. Without loss of generality set both agents to be inside corridor 1, i.e., the lower corridor.

Assume that the goals for the agents are symmetrically placed at the end of each side of the corridor. Then the shortest path between the two goal points through the corridors is x, as illustrated in Fig. 4. If the corridors are too narrow, then one of the paths will go through the current configurations of robots A and B. Assume that the distance between the goal G_B and q_A is ε_A and the distance between the goal G_A and q_B is ε_B along the path that goes through corridor 1 (the lower corridor). Then the lengths of the shortest paths for the robots to reach their goals via the corresponding homotopic paths can be computed as follows:

$$P_1^A = x - \varepsilon_A,\ P_2^A = x + \varepsilon_A, \qquad P_1^B = x - \varepsilon_B,\ P_2^B = x + \varepsilon_B,$$

where P_i^X corresponds to the length of the shortest path for robot X from its current configuration q_X to its goal G_X via corridor i.

The proposed approach also considers an interaction cost along each action for every agent. The interaction cost of an action is 0 if there is no other agent occupying the corresponding path given the latest observation. If there is an agent occupying the path, then the interaction cost is computed as follows:

$$I_i^A = 1 - \frac{\text{distance between A and B along } \pi_i}{\text{length of } \pi_i} \tag{1}$$

The reasoning behind this definition is that agents closer to the current position of an agent should incur a higher interaction cost. Then for the above scenario the interaction costs are:

$$I_1^A = \frac{\varepsilon_B}{x - \varepsilon_A},\ I_2^A = 0, \qquad I_1^B = \frac{\varepsilon_A}{x - \varepsilon_B},\ I_2^B = 0.$$

Then the proposed cost function for actions is $C_i^X = P_i^X(1 + 2 \cdot I_i^X)$, which translates to the following costs in the above scenario:

$$C_1^A = x - \varepsilon_A + 2 \cdot \varepsilon_B,\ C_2^A = x + \varepsilon_A, \qquad C_1^B = x - \varepsilon_B + 2 \cdot \varepsilon_A,\ C_2^B = x + \varepsilon_B.$$

Then, note that in order for A to select action 1 it has to be the case that:

$$C_1^A = x - \varepsilon_A + 2 \cdot \varepsilon_B < x + \varepsilon_A = C_2^A \Rightarrow \quad A \text{ selects corridor 1 iff: } \varepsilon_B < \varepsilon_A \quad (2)$$

Similarly for robot B to select action 1 it has to be the case that:

$$C_1^B = x - \varepsilon_B + 2 \cdot \varepsilon_A < x + \varepsilon_B = C_2^A \Rightarrow \quad B \text{ selects corridor 1 iff: } \varepsilon_A < \varepsilon_B \quad (3)$$

From Eqs. 2 and 3 it becomes apparent that the agents are not able to simultaneously pick the same corridor given the above definitions for the interaction cost and the overall cost functions. The agent who is farther away from its goal will have to pick the other homotopy class.

The entire above discussion was based on the assumption that the goal locations of the two agents were symmetrical relative to the corridors, i.e., the length of the path connecting the agents that goes through corridor 1 is the same as the length of the path through corridor 2. If the goals are not symmetrical, then instead of a common path length of x, the initial path costs P_i^X should include different lengths x_1 and x_2 for the connections of the goals via corridor 1 and 2 respectively. Then, the cost of actions should be defined in a general manner: $C_i^X = P_i^X(1 + \alpha \cdot I_i^X)$ for a constant α, which will depend on the relative difference $\Delta x = |x_1 - x_2|$. This is information, however, that is not available to the robots, since it requires knowledge of the goals for the other agents.

In practice, using the value $\alpha = 2$ as was done in this section, results in good performance in the classification of different homotopic paths in terms of their interaction cost. So, in the context of the replanning framework in order to replace the greedy choice, a "k-best" choice is used. First, compute the k-shortest paths that belong to k different homotopy classes. Then, for each one of these paths, compute their costs according to the definition of C_i^X, where the interaction cost is computed according to Eq. 1. The action with minimum cost both minimizes distance from the goal as well as interaction with other agents. The above "k-best" strategy is superior to the "greedy" strategy of always selecting the shortest path, since it allows a robot to consider multiple alternative choices as well as interactions with neighbors. In this manner, it provides a resolution to the basic "corridor" challenge under the assumption that the goals of the two agents are symmetric.

Minimum Conflict (`MC`) **Path**: The "k-best" strategy was able to avoid deadlocks as long as the total number of simple homotopy classes did not significantly exceed k, where simple homotopy classes correspond to those that do not include loops. When this property is true, then the strategy described in the previous section results in the selection of paths which allow the team to make progress overall. Even in relatively simplistic scenes, however, the number of homotopy classes can quickly become large. This introduces a computational challenge, since the $k + 1$th homotopy class corresponds to an increasingly longer path, which translates to a longer search time on the underlying roadmap. To keep the proposed method computationally

effective, however, it is important to keep a small planning cycle and perform each path computation as fast as possible.

In order to address this issue, the value of k is kept relatively small, and to accommodate the potential lack of a desirable path, the current work proposes that the "minimum-conflict path" should always be included as an available action to the agents. To compute such a path, each agent a_i considers the current set of configurations for the agents it can observe: $\{q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_m\}$. For each one of those configurations q_j, agent a_i marks edges in the roadmap that intersect q_j. Edges that are marked then have their weights inflated by a large amount, effectively removing them from consideration during the heuristic search to find the shortest path on the roadmap from q_i to q_i^G. This means that the heuristic search process will first return the shortest path that does not collide with any agents. If no such path exists, then one which collides only with one agent will be returned and so on.

The inclusion of such paths in the set of available strategies results in methodologies that always solve challenges where the "greedy" or the "k-best" method failed. Interestingly, a strategy which only considers the "minimum conflict" action, constructed at each replanning cycle using the process described above, is also able to always solve all the challenges considered in the accompanying simulations.

Even so, the resulting paths may not be as desirable when all the agents follow their minimum conflict action. As shown in Fig. 5, this action may be significantly longer than the shortest path to the goal. Thus, it is interesting to consider the combination of the "k-best" strategy with the "minimum conflict" one. In this case, the process works as follows: first, a homotopy class computation algorithm is used to extract the k-shortest paths that belong to k different homotopy classes. Next, the "minimum-conflict" action is computed. For each one of the above $k + 1$ paths, their costs are computed according to the definition of C_i^X, where the interaction cost is from Eq. 1. Finally, the action with the minimum cost is returned.

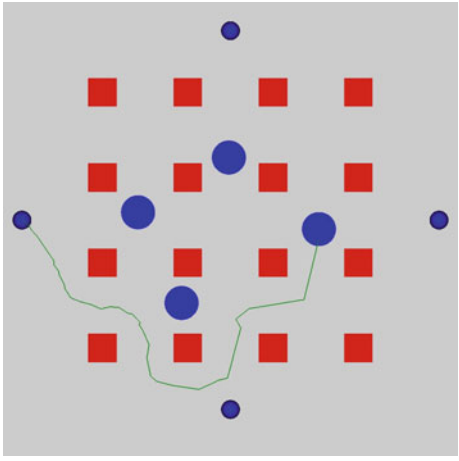


Fig. 5 A "minimum-conflict" path computed for the right-most agent for a goal to the left. The shortest path without conflicts is returned. For a large distribution of agents, the "minimum-conflict" path will typically intersect some agents

This "deterministic" approach for combining the agent's greedy choices, i.e., k-shortest paths, and the safe/conservative choice, i.e., the minimum conflict path, takes advantage of the process for evaluating interaction costs. It allows agents to sometimes select one of the shortest paths, even if they conflict with other agents, as long as these paths are significantly shorter than the minimum conflict path and do not overlap with other agents early on.

A Probabilistic Selection Strategy To allow some adaptability to the choices of other agents, this work considers an online learning method to select the appropriate strategy out of the following: (a) the "minimum conflict" (the shortest path with the least amount of agent interaction), and (b) a greedy strategy, where this work considers two possible alternatives for the greedy strategy—returning the shortest path ignoring other agents or, returning the action selected by the "k-best" strategy. The objective of this approach is to learn during the execution of a path whether it is better to play the conservative/"minimum conflict"strategy or the greedy alternative, given the cost that it experiences for the outcomes of these strategies over time.

The learning approach corresponds to the Polynomial Weights method, which applies regret minimization [21, 22]. It begins by assigning uniform weights on the two strategies: $w^{min\_conflict} = w^{greedy} = 1$. Then, when the agent must choose an action, one of the strategies is chosen at random proportionally to their weights, i.e.,

$$Pr(\text{"minimum-conflict"}) = \frac{w^{min\_conflict}}{w^{min\_conflict} + w^{greedy}}.$$

During each planning cycle, the method updates these weights by calculating a loss value for each one of them: $l^{min\_conflict}$, l^{greedy}, in hindsight, i.e., assuming that all the other agents would have acted the same way, the method computes a value that corresponds to the regret of choosing that value. Given the other agents' motion, one of the two pure strategies would have performed better. This action has low regret and its weight is not reduced, while the worse performing strategy incurs regret, and thus receives a lowered weight. The implementation of Polynomial Weights in the context of this challenge implies a loss computation as follows:

$$l_i = \frac{C_i - min_i(C_i)}{max_i(C_i) - min_i(C_i)}$$

The term C_i again corresponds to the weighted interaction cost. The weights are then updated according to the following rule and the computed loss value: $w_i = w_i \cdot (1 - \eta \cdot l_i)$. This means that the action with the highest weighted cost in hindsight gets its weight reduced by η, while the other action is not penalized. A value of $\eta = 0.2$ was used for the simulations presented here.

The Polynomial Weights method has several advantages. First, it does not require knowledge of the other agent's utilities and requires no information to be passed from the other agents. Furthermore, as the weights are learned, the expected utility

is guaranteed to be within a bound of the best pure strategy [21, 22]. Lastly, it allows a high degree of adaptability to changing conditions, as large regret costs will be quickly accumulated for choosing a sub-optimal strategy.

4 Simulations

Each of the strategies presented in the previous sections was evaluated experimentally using appropriate simulation software [17]: shortest path (`Greedy`), K-best (`KBest`), Minimum-conflict (`Min Conf`), Deterministic combination of `Min Conf` and `KBest` (`Determ`), and Polynomial-Weights with (`Greedy`) (`PWGreedy`) and with `KBest` (`PWBest`).

The metrics used were average completion time in seconds for all agents and average length of the solution path for all agents. Evaluating the average experimental solution time provides a good measure of the performance of the method, as it directly indicates how much progress agents are making towards their goals. The purpose of examining the average path length is to have some measurement of how much "effort" an agent must spend to achieve its desired solution time.

Each agent had a physical radius of 18 cm and a sensing radius of 200 cm. A visualization of the environments used in the experiments is shown in Fig. 6. In the Grid environment, the block obstacles measured 30 $\times$ 30 cm, and were placed so as to create corridors with width 40 cm. In the Random Obstacle environment, the obstacles were cylinders with variable radius from 10 to 40 cm.

A centralized optimal path planner is not compared against in the performance section, as it would remove the "minimum information" requirement while only providing a comparison point for small numbers of agents. Scenes without enough agents in them would not introduce many conflicts, so the proposed approaches do not provide benefit over using existing reactive obstacle avoidance.

Evaluating Validity: The current work can only provide guarantees on conflict avoidance in scenarios such as the one presented in Sect. 3: Minimizing Interaction Cost. Accordingly, the experiments begin with a simple corridor setup, with two agents having symmetrical goals and attempting to reach opposite sides of the corridor. The purpose of such a simplistic setup is to find whether the proposed methods,

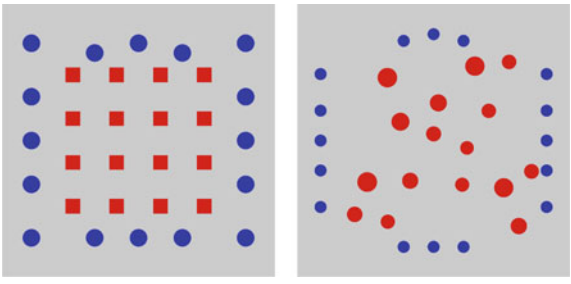


Fig. 6 The environments used to evaluate the proposed methods. The *blue* disks are the agent's initial positions, when they are not randomized

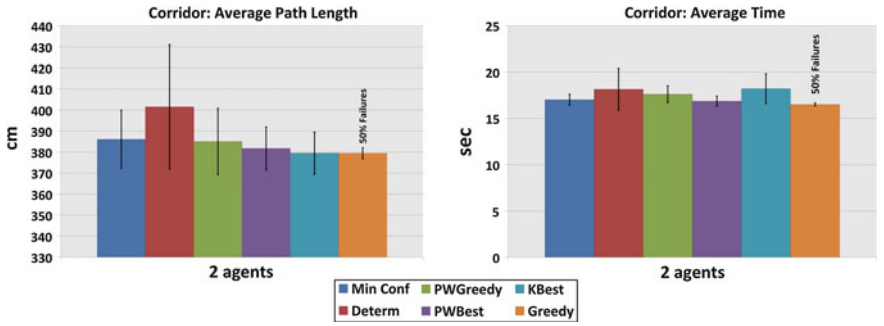


Fig. 7 Validation results in the corridor scenario of Fig. 1

including the `Greedy` approach, are able to solve simple congestion problems as well as testing the validity of the framework in simulation. The results are averaged over 10 runs, with the average path lengths and solution times shown in Fig. 7.

Although `Greedy` always had the lowest averages, it failed to solve even a simple deconfliction problem such as this 50 % of the time. This is again due to the fact that no other paths are considered by the agent. All of the other strategies were able to solve the corridor problem without a single failure.

Evaluating Performance: The performance of the four methods is evaluated using two environments, the grid environment and a random obstacle environment as shown in Fig. 6. Since the `Greedy` strategy failed to consistently solve the corridor problem, it is omitted from the rest of the experiments. An important observation of the `KBest` strategy is for small values of k, and for large numbers of homotopy classes, it is possible for the strategy to become deadlocked/livelocked. Such was the case in the grid and random environments, so accordingly the `KBest` strategy is no longer considered in further experiments.

Agents are given a pseudo-random start location, and a fixed goal location, with the intention of having agents swap locations with one another, which promotes conflicts and congestion in the environment. The results for the grid environment are averaged over 10 runs and presented in Fig. 8.

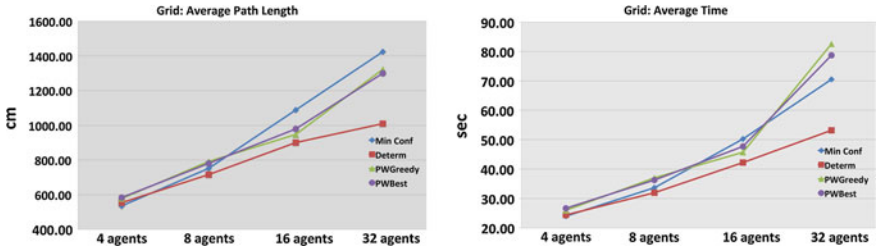


Fig. 8 Results for randomly selected starting positions in the grid environment

The deterministic approach, `Determ`, which considers both the "minimum-conflict" and the "k-best" strategies, always selects the action that minimizes the proposed interaction cost. In the Grid environment, `Determ` outperformed the other approaches. This makes sense since these experiments were homogeneous (all agents used the same approach), so the adaptive strategies could not take advantage of any differences in strategies (such as learning to follow a greedy strategy against agents that play conservatively and follow a minimum-conflict strategy).

A set of experiments was conducted in the random obstacle scene, however, the results showed that all the approaches performed at roughly the same level. The explanation for this is that the random placement of obstacles, combined with a larger workspace, did not cause a constrained enough environment, hence there were not frequent conflicts between agents. This allowed agents to consider a larger set of possible actions that are conflict-free, so each of the strategies presented were able to provide an equivalent quality solution.

Evaluating Performance Without Reciprocal Velocity Obstacles A requirement of the Reciprocal Velocity Obstacle (RVO) method is that agents are able to sense the velocities of neighboring agents. In order to study the viability of the proposed methods without relying on sensing velocities, a "lite" version of the methods are evaluated, where the RVO method is replaced with a simplistic, position-based approach (agents stop moving in a particular direction if this direction brings it too close to another agent).

To get a better idea of how well the proposed methods perform, and to serve as a comparison point, a "straw-man" type algorithm was implemented: when an agent finds its currently selected path results in a conflict, it randomly selects a different, potentially viable path. This algorithm is presented in the results as the "KRand" algorithm. The results are presented in Fig. 9.

All of the methods, except for KRand, were able to solve the problems a 100 % of the time. The comparison method, KRand, provided competitive solution path lengths, at the expense of much longer solution times, as well as a 40 % failure rate in the 32 agent scenario. Although `Min Conf` always solved the problems, both `Determ` and `PWBest` outperformed it in solution time and solution length.

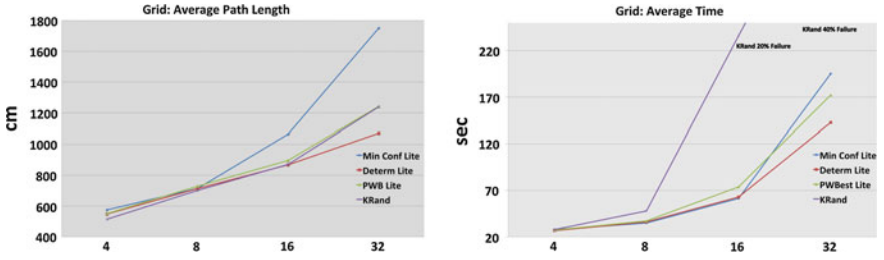


Fig. 9 Results for randomized grid using the "lite" versions of the algorithms. The Reciprocal Velocity Obstacle collision avoidance method is replaced with a simple, position-based method

Fig. 10 The average path length and average time to finish for simulations using the polynomial weight with greedy (`PWGreedy`) and with k-best selection (`PWBest`)

Evaluating Scalability In these experiments the start location of the agents were set to be symmetrical, so as to promote conflicts and congestion quickly. The results are averaged over 15 different runs and are shown in Fig. 10. The purpose of this set of experiments was to evaluate the scalability of the adaptive-strategies, `PWGreedy` and `PWBest`, as both of these approaches utilize the other deconfliction methods, and are consequently the most computationally complex.

The results show that for increasingly larger number of agents, the average solution time and the average path lengths for the methods scales sublinearly.

Heterogeneous Setups A set of experiments was conducted among heterogeneous agents in the grid environment, where 7 agents were assigned the "minimum-conflict" strategy and 1 agent was assigned the `PWBest` strategy. The idea here was to examine the probabilistic learning algorithm, `PWBest`, and see if it was able to adapt its weights according to the strategies the other agents were playing. Interestingly, over a course of 5 separate runs, `PWBest` selected the "k-best" strategy 65 % of the time on average. Since the "minimum-conflict" agents were actively attempting to avoid interaction with other agents, it makes sense that the `PWBest` agent is able to be more "greedy" in its selection of paths.

Carrying on with this line of thought, another set of experiments was run where 4 agents were given the pure "greedy" strategy, and the other 4 agents ran `PWBest`. In this case, the `PWBest` agents adapted and chose to select the "k-best" strategy only 41 % of the time. Since the 4 purely greedy agents caused a deadlock in the center of the environment, the `PWBest` agents had to adapt and select the safer "minimum-conflict" strategy more often. Together these results seem to show promise for the adaptability of the learning strategy, as well as motivating its use over the "deterministic" strategy in the general case, since their path length is equivalent in homogeneous setups. A video providing a qualitative description of the performed experiments can be found at: http://www.cs.rutgers.edu/~kb572/dars2014.mp4.

5 Discussion

The proposed framework brings together path planning primitives, such as search-based methods for computing paths in different homotopy classes [3] and sampling-based motion planners for computing roadmaps [14], reactive obstacle avoidance methods [27, 30] as well as game theoretic and learning tools [22] to provide an algorithmic framework capable of computing acceptable solutions to motion coordination challenges in a decentralized, communication-less way.

There are many interesting future directions for this line of research. This includes the evaluation of approaches on dynamical systems as well as more complex environments and removing the "garage" assumption from the framework. This would require agents to continue reasoning about their observed states, potentially adapting a "passive" mode to more easily allow other agents through their goal positions. Additional experimentation of the adaptive, learning methods in a larger set of heterogeneous setups is interesting, as well as imposing a stricter sensing range on the agents. Furthermore, it is important to analyze the conditions under which the current framework, in particular the consideration of the "minimum-conflict" path, is able to guarantee that the robots are free of deadlocks and livelocks, using tools that have been developed towards this direction [20].

References

1. Bekris, K.E., Grady, D.K., Moll, M., Kavraki, L.E.: Safe distributed motion coordination for second-order systems with different planning cycles. IJRR **31**(2) (2012)
2. Bhattacharya, S., Kumar, V., Likhachev, M.: Search-based path planning with homotopy class constraints. In: Third Annual Symposium on Combinatorial Search (2010)
3. Bhattacharya, S., Likhachev, M., Kumar, V.: Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In: RSS (2011)
4. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Auton. Robots **33**(3), 273–290 (2012)
5. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. IJRR 17(7) (1998)
6. Fraichard, T., Delsart, V.: Navigating dynamic environments with trajectory deformation. J. Comput. Inf. Technol. **17**(1) (2009)
7. Green, C., Kelly, A.: Toward optimal sampling in the space of paths. In: ISRR (2007)
8. Hatcher, A.: Algebraic Topology. Cambridge University Press (2002)
9. Hauser, K.: Adaptive time stepping in real-time motion planning. In: Algorithmic Foundations of Robotics IX, pp. 139–155. Springer (2011)
10. Hauser, K.: Minimum constraint displacement motion planning. In: RSS (2013)
11. Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: ICRA. Anchorage, AK (2010)
12. Jaillet, L., Siméon, T.: Path deformation roadmaps. In: Workhop on the Algorithmic Foundations of Robotics (WAFR) (2006)
13. Kaminka, G., Erusalimchik, D., Kraus, S.: Adaptive multi-robot coordination: a game-theoretic perspective. In: ICRA (2010)
14. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. In: IJRR (2011)

15. Karamouzas, I., Geraerts, R., Overmars, M.: Indicative routes for path planning and crowd simulation. In: Foundations of Digital Games (FDG), pp. 113–120 (2009)
16. Kimmel, A., Dobson, A., Bekris, K.E.: Maintaining team coherence under the velocity obstacle framework. In: Autonomous Agents and Multiagent Systems (AAMAS) (2012)
17. Kimmel, A., Dobson, A., Littlefield, Z., Krontiris, A., Marble, J., Bekris, K.E.: Pracsys: an extensible architecture for composing motion controllers and planners. In: SIMPAR (2012)
18. Knepper, R.A., Mason, M.T.: Path diversity is only part of the problem. In: ICRA (2009)
19. Knepper, R.A., Rus, D.: Pedestrian-inspired sampling-based multi-Robot collision avoidance. In: IEEE RO-MAN, pp. 94–100. IEEE, Paris, France (2012)
20. Knepper, R.A., Rus, D.: On the completeness of ensembles of motion planners for decentralized planning. In: ICRA. Karlsruhe, Germany (2013)
21. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**, 212–261 (1994)
22. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press (2007)
23. Petti, S., Fraichard, T.: Partial motion planning framework for reactive planning within dynamic environments. In: ICINCO, pp. 199–204 (2005)
24. Qutub, S., Alami, R., Ingrand, F.: How to solve deadlock situations within the plan-merging paradigm for multi-Robot cooperation. IROS **3**, 1610–1615 (1997)
25. Shi, D., Collins, E.G., Donate, A., Liu, X., Goldiez, B., Dunlap, D.: Human-aware Robot motion planning with velocity constraints. In: IEEE International Symposium on Collaborative Technologies and Systems, pp. 490–497 (2008)
26. Sisbot, E.A., Marin-Urias, L.F., Alami, R., Siméon, T.: A human-aware mobile Robot motion planner. IEEE Trans. Robot. **23**(5), 874–883 (2007)
27. Snape, J., van Den Berg, J., Guy, S., Manocha, D.: The hybrid reciprocal velocity obstacle. IEEE Trans. Robot. **27**(4), 696–706 (2011)
28. Thompson, S., Horiuchi, T., Kagami, S.: A probabilistic model of human motion and navigation intent for mobile Robot path planning. In: ICARA (2009)
29. van Den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of individual agents in crowded environments. In: I3D (2008)
30. van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
31. van den Berg, J., Snape, J., Guy, S., Manocha, D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In: ICRA (2011)
32. Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A., Srinivasa, S.: Planning-based prediction for pedestrians. In: IROS (2009)

Scalable Formation Control of Multi-robot Chain Networks Using a PDE Abstraction

Karthik Elamvazhuthi and Spring Berman

Abstract This work investigates the application of boundary control of the wave equation to achieve leader-induced formation control of a multi-robot network with a chain topology. In contrast to previous related work on controlling formations of single integrator agents, we consider a model for double integrator agents. For trajectory planning, we use the flatness based method for assigning trajectories to leader agents so that the agents' trajectories and control inputs are computed in a decentralized way. We show how the approximation greatly simplifies the planning problem and the resulting synthesized controls are bounded and independent of the number of agents in the network. We validate our formation control approach with simulations of 100 and 1000 agents that converge to configurations on three different type of target curves.

Keywords Formation control · Boundary control · Wave equation · Flatness-based trajectory planning · Chain networks · Scalable control

1 Introduction

A considerable amount of effort has been applied in recent years to problems of achieving consensus, coverage, task allocation, and coordinated motion in multi-robot systems. In particular, certain multi-robot applications will require formation control of the robots to positions along a specified closed or open curve within a certain amount of time. For instance, the curve could lie along an object to be transported or a structure to be monitored, or it could contain target formations or flocking trajectories for aerial vehicles and spacecraft. Moreover, formation control provides useful benchmarks for investigating the range of coordinated behaviors that can be

K. Elamvazhuthi · S. Berman (✉)
School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, USA
e-mail: Spring.Berman@asu.edu

K. Elamvazhuthi
e-mail: karthikevaz@asu.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_25

achieved under the constraints that are typical to multi-robot systems. These constraints include unreliable or absent communication and global information, limited resources for sensing and computation, and the presence of unpredictable environmental disturbances. In addition, control schemes for coordinating large multi-robot systems should be scalable to arbitrary robot population sizes.

The virtual structure method is a well-known approach to formation control [18]. It is based on a combination of consensus and graph rigidity concepts, in which agents know the target distance to be maintained from their neighbors on a graph that defines their interaction topology. While maintaining these inter-agent distances, they must also simultaneously reach consensus on the center of the formation. Other approaches to this problem include potential field methods [11] and control Lyapunov functions for multi-agent coordination [15].

Alternative approaches to multi-robot formation control have been derived from partial differential equation (PDE) models of the system. The applicability of many of these PDE models is based on the fact that finite difference approximations of differential operators on continuous domains have the same structure as analogous operators defined on graphs (e.g., the Laplacian) and also provide intuition on the use of analogous operators on graphs such as advection-based coordination [5]. In [3], finite difference approximations of PDEs used in image processing are applied to a cooperative boundary coverage problem. The work in [17] used another numerical solution method for PDEs, the smoothed particle hydrodynamics method, to model finite-sized robots as an incompressible fluid, incorporating nonholonomic constraints and obstacle avoidance. PDE models were used in [20] to analyze the string stability of large vehicle platoons, and PDE approximations of vehicle platoons were applied in [2] to study the scaling behavior of system stability margins with the number of agents. Linear and nonlinear advection-diffusion models are used in [8] to enable deployment of multiple agents into formations using a boundary control methodology. Similar work was done by [13] on planar deployments of multiple agents using flatness-based trajectory planning of the Burgers' equation. Fluid flow models are also extensively used in traffic flow problems [7]. Closely related to partial differential equations is the concept of partial difference equations on graphs, which has also been a subject of a number of studies. Spatially invariant systems are another type of infinite-dimensional approximation of large-scale networks. These have proven to be quite insightful in understanding scaling laws and structure-dependent performance limitations of large vehicle networks [1, 10].

In this work, we address a formation control problem for a multi-agent network with an undirected chain graph topology. None of the agents have communication capabilities, and two of the agents (the *leader agents*) have global position information while the remaining robots (the *follower agents*) have only local sensing and cannot measure their global positions. Each agent's motion is governed by double integrator dynamics, which integrate force actuation as the local control parameter. For agents with double integrator dynamics, the wave equation serves as a useful abstraction. This is in contrast with the single integrator agent models studied in [8, 13], which employed advection-diffusion equation abstractions for formation control. We demonstrate that trajectory planning based on the wave equation can

implicitly account for delays in system controllability that are inherent to this type of model. In doing so, our approach enables the synthesis of bounded control inputs that drive chain networks with arbitrarily large numbers of agents to configurations on target open and closed curves. For our trajectory planning approach, we use the flatness based method, a well-known method in the context of finite-dimensional systems [21]. Namely, using the so-called *flat output*, we achieve an explicit parametrization of the system states and control input. Since its introduction, the method has been extended to infinite-dimensional systems as well [12, 19]. This approach serves as a useful alternative to optimal control methods, which require numerous cycles of integration and, in the case of PDE models, lead to ill-conditioning issues arising from numerical discretization.

2 Problem Formulation

We consider a group of N non-communicating agents that move in the space $\mathbb{R}^n$, $n \in \{1, 2, 3\}$. The position and control input of agent j at time t are denoted by $\mathbf{z}_j(t) = [z_j^1(t) \ \ldots \ z_j^n(t)]^T \in \mathbb{R}^n$ and $\mathbf{u}_j(t) = [u_j^1(t) \ \ldots \ u_j^n(t)]^T \in \mathbb{R}^n$, respectively. We assume that agent j can measure its distance from two other agents $j-1$ and $j+1$ at all times, or in other words, that agent j is *connected* to agents $j-1$ and $j+1$. The agent interconnection topology forms a one-dimensional undirected chain graph. No agent can measure its global position $\mathbf{z}_j(t)$ except for agents 1 and N, which we call the *leader agents*. The positions of the leader agents evolve according to specified trajectories:

$$z_1^i(t) = u_1^i(t), \quad z_N^i(t) = u_N^i(t), \quad i = 1, \ldots, n. \tag{1}$$

The dynamics of the *follower agents* $j = 2, \ldots, N-1$ are given by double-integrators with control inputs on their acceleration:

$$u_j^i(t) = \frac{d^2 z_j^i(t)}{dt^2} = c^2 \left[(z_{j+1}^i(t) - z_j^i(t)) - (z_j^i(t) - z_{j-1}^i(t)) \right] + f_j^i, \quad i = 1, \ldots, n, \tag{2}$$

where c and f_j^i are constants.

The control objective is to drive the agents' positions to points along a target open or closed curve, $\gamma : [0 \ 1] \to \mathbb{R}^n$, at equilibrium. We will show that we can achieve this objective by designing the leader agents' position control inputs, $\mathbf{u}_1(t)$ and $\mathbf{u}_N(t)$, and the follower agents' constant acceleration inputs f_j^i. Toward this end, we define $h = 1/N$ and rewrite Eq. (2) as

$$u_j^i(t) = \frac{d^2 z_j^i(t)}{dt^2} = (ch)^2 \frac{(z_{j+1}^i(t) - 2z_j^i(t) + z_{j-1}^i(t))}{h^2} + f_j^i, \quad i = 1, \ldots, n. \tag{3}$$

As $N \to \infty$, Eq. (3) converges to n one-dimensional partial differential equations (PDE's) that evolve in time over a continuous spatial domain, which we normalize to the interval [0 1]. The agent population is represented as a continuum with a spatial distribution in dimension i given by $z^i(x,t), x \in [0\ 1]$. Note that the positions of the leader agents are then $z^i(0,t)$ and $z^i(1,t)$, which are defined at the endpoints of the domain. We assume zero initial conditions, so the entire agent population begins at rest at the origin. The spatiotemporal evolution of $z^i(x,t)$ is then governed by the following set of *n forced wave equations* with time-dependent Dirichlet boundary conditions:

$$\frac{\partial^2 z^i(x,t)}{\partial t^2} = (ch)^2 \frac{\partial^2 z^i(x,t)}{\partial x^2} + f^i(x) \tag{4}$$

$$z^i(0,t) = u_0^i(t), \quad z^i(1,t) = u_N^i(t) \tag{5}$$

$$z^i(x,0) = 0, \quad \frac{dz^i(x,0)}{dt} = 0. \tag{6}$$

The time-independent source function $f^i(x)$ can be designed to specify the target curve γ to which the agent positions converge at equilibrium. The product ch defines the speed of wave propagation over the domain.

We will use the nonhomogeneous boundary value problem Eqs. (4)–(6) to plan trajectories for the multi-agent system. We will discuss how this continuous approximation simplifies the planning problem and takes into account some inherent limitations that are not obvious in the original discrete control system Eqs. (1) and (2).

3 Limitations on Controllability

The continuous PDE approximation, which is an infinite-dimensional system, shares certain controllability limitations with the original system, which has a finite-dimensional state space. If a finite-dimensional system is controllable at a particular time, the Kalman rank condition can be used to show that the system is controllable at any time. This result implies that leader-follower protocols on grid networks are controllable for any number of agents in the network. Analysis of the controllability gramian shows that as the agent population increases, there is a rise in the minimum energy required to drive the network to a chosen target state [16, 23]. However, there is a more fundamental problem in controlling large networks of double-integrator agents than the need for a correspondingly large amount of control energy. This problem is the constraint on the minimum time needed to drive the network to a target state.

In contrast to finite-dimensional systems, infinite-dimensional systems have much more varied notions of controllability. In particular, for a system described by a wave equation whose Laplacian operator has coefficient k^2, a minimum time of $T = 2/k$ is required to attain exact controllability of the system [9]. This is due to the finite speed

of propagation of a wave over a one-dimensional spatial domain. Even if a leader agent introduces an infinite amount of control effort, information cannot travel faster than this speed through the network. Moreover, high-frequency disturbances over the network will take even longer to stabilize due to the relatively lower speed of the wave packets. Conversely, systems described by the heat equation, which can be considered to be an infinite-dimensional version of the single-integrator based Laplacian models [4], are not subject to a delay in controllability. These models have an infinite speed of information propagation over the domain, and hence controllability at any time, albeit only approximately in the continuous case.

Consequently, while a single-integrator agent network with a grid topology can be controlled to any state at any time, the minimum time T needed to control a double-integrator network with continuous approximation Eq. (4) increases with the number of agents N as $T = 2/(ch) = 2N/c$. For large N, it therefore takes a long time for the control effort of a leader agent to propagate through the entire network, resulting in a large set of states that are unreachable for times $t \leq T$. This issue can be identified with the problem of numerically approximating optimal controllers of the wave equation by constructing optimal controllers of the corresponding semi-discrete system [24]. When the controls are constructed in this manner, they diverge as the number of mesh points (equivalent to agents in our case) increases, in spite of the convergence of the discrete model to the continuous model. This divergence has been attributed to (a) the finite time needed for controllability, and (b) the mismatch between the wave speeds of high-frequency perturbations in discrete and continuous media [24]. Owing to the richer dynamics of the semi-discrete system, the minimum time required to reach a target state might be even higher than that indicated by the continuum approximation.

4 Trajectory Planning

In this section, we show that trajectory planning based on the wave equation can implicitly account for the delay in system controllability that is described in Sect. 3. In doing so, this approach enables the synthesis of bounded control inputs that drive chain networks with arbitrarily large numbers of agents to target configurations.

We modify the follower agent control inputs defined in Eq. (2) by scaling the constant c by the agent population N, which changes the coefficient c^2 to $(cN)^2 = (c/h)^2$. Then, as $N \to \infty$, Eq. (2) converges to the wave equation in Eq. (4) with the coefficient $(ch)^2$ replaced by c^2. The corresponding speed of wave propagation is c, which is independent of the number of agents. The control effort per agent defined in Eq. (2) remains bounded as $N \to \infty$ due to the convergence of the semi-discrete system to its continuous approximation. Strictly speaking, this argument requires strong convergence, which can be easily achieved by introducing a small amount of damping in the system using velocity-based compensation [14].

To simplify the construction of the control inputs, we decompose the boundary value problem Eqs. (4)–(6), with $(ch)^2$ replaced by c^2, into two components. This

decomposition is possible because the wave equation Eq. (4) is a linear PDE. The first component is a boundary value problem for an *unforced wave equation* (note that the superscript i has been suppressed to simplify the notation):

$$\frac{\partial^2 z(x,t)}{\partial t^2} = c^2 \frac{\partial^2 z(x,t)}{\partial x^2} \tag{7}$$

$$z(0,t) = 0, \quad z(1,t) = u_{1a}(t) \tag{8}$$

$$z(x,0) = 0, \quad \frac{\partial z(x,0)}{\partial t} = 0 \tag{9}$$

The second component is a boundary value problem for a *forced wave equation*:

$$\frac{\partial^2 z(x,t)}{\partial t^2} = c^2 \frac{\partial^2 z(x,t)}{\partial x^2} + f(x) \tag{10}$$

$$z(0,t) = u_0(t), \quad z(1,t) = u_{1b}(t) \tag{11}$$

$$z(x,0) = 0, \quad \frac{\partial z(x,0)}{\partial t} = 0 \tag{12}$$

Here, $u_{1a}(t)$ drives the system to equilibrium, while $u_0(t)$ and $u_{1b}(t)$ shift the datum of the solution depending on the desired target state.

4.1 Unforced Wave Equation Component

Following the approach of [22] for flatness based trajectory generation, we take the Laplace transform of Eq. (7) in the time variable and obtain

$$s^2 Z(x,s) = c^2 \frac{d^2 Z(x,s)}{dx^2} \tag{13}$$

The general solution of this equation is

$$Z(x,s) = A(s)\cosh\left(\frac{xs}{c}\right) + B(s)\sinh\left(\frac{xs}{c}\right) \tag{14}$$

where $A(s)$ and $B(s)$ are arbitrary functions of s. Applying the boundary condition $Z(0,s) = 0$ to Eq. (14), we find that $A(s) = 0$. Now define the function $r(t) = \partial z(0,t)/\partial x$. The Laplace transform of this function is $R(s) = dZ(0,s)/dx$, which is the derivative of Eq. (14) with $x = 0$. This derivative is $R(s) = B(s)\frac{s}{c}\cosh(0) = B(s)\frac{s}{c}$, which yields $B(s) = R(s)\frac{c}{s}$. Let $y(t)$ denote the flat output, and define its Laplace transform as $Y(s) = R(s)\frac{c}{s}$. Then, Eq. (14) becomes

$$Z(x,s) = Y(s)\sinh\left(\frac{xs}{c}\right) = \frac{1}{2}Y(s)e^{xs/c} - \frac{1}{2}Y(s)e^{-xs/c} \tag{15}$$

Applying the boundary condition $Z(1, s) = U_{1a}(s)$ to Eq. (15), we obtain

$$U_{1a}(s) \;=\; Y(s)\sinh\left(\frac{s}{c}\right) \;=\; \frac{1}{2}Y(s)e^{s/c} - \frac{1}{2}Y(s)e^{-s/c}. \tag{16}$$

Taking the inverse Laplace transform of Eqs. (15) and (16) yields a parametrization of the state and input trajectories in terms of the output:

$$z(x,t) = \frac{1}{2}y\left(t + \frac{x}{c}\right) - \frac{1}{2}y\left(t - \frac{x}{c}\right) \tag{17}$$

$$u_{1a}(t) = \frac{1}{2}y\left(t + \frac{1}{c}\right) - \frac{1}{2}y\left(t - \frac{1}{c}\right) \tag{18}$$

The input is therefore defined by the output values at times $t - \frac{1}{c}$ and $t + \frac{1}{c}$. Since time must be nonnegative, $(t - \frac{1}{c}) \geq 0$, which implies that $(t + \frac{1}{c}) \geq \frac{2}{c}$. Hence, a minimum time of $t = 2/c$ is needed to drive the system from its initial state to its final state. This is consistent with the controllability results for the wave equation that were discussed in Sect. 3.

Let T be the time at which the system is to be driven to the target state. We define the functions $g(x) = y(T + \frac{x}{c})$ and $h(x) = y(T - \frac{x}{c})$ and denote the target state and its desired time derivative by $z^*(x)$ and $z_t^*(x)$, respectively. We obtain the following expressions for $z^*(x)$ and $z_t^*(x)$:

$$z^*(x) = z(x,T) = \frac{1}{2}g\,(x) - \frac{1}{2}h\,(x) \tag{19}$$

$$z_t^*(x) = \frac{\partial z(x,T)}{\partial t} = \frac{1}{2c}\frac{dg\,(x)}{dx} + \frac{1}{2c}\frac{dh\,(x)}{dx} \tag{20}$$

Solving Eqs. (19) and (20) for $g(x)$ and $h(x)$ yields:

$$g(x) = c\int_0^x z_t^*(\sigma)d\sigma \;+\; z^*(x) \tag{21}$$

$$h(x) = c\int_0^x z_t^*(\sigma)d\sigma \;-\; z^*(x) \tag{22}$$

We set $y(t) = 0$ for $t \leq T - \frac{2}{c}$. Then, a boundary control trajectory $u_{1a}(t)$ that drives system to the desired target state can be constructed as:

$$u_{1a}(t) = \begin{cases} 0, & t \in [0, T - \frac{2}{c}) \\ \frac{1}{2}h(cT - 2 - ct), & t \in [T - \frac{2}{c}, T - \frac{1}{c}) \\ \frac{1}{2}g(ct - cT + 2), & t \in [T - \frac{1}{c}, T] \end{cases} \tag{23}$$

4.2 Forced Wave Equation Component

The boundary control inputs Eq. (11) in the forced wave equation Eq. (10) are defined as $u_0(t) = z^*(0)$ and $u_{1b}(t) = z^*(1)$ for $t \geq T$. These control inputs drive the leader agents to their target final locations and anchor them there. Their design accounts for the fact that the left boundary in the unforced wave equation Eq. (7) is always fixed at zero. Using the superposition principle, the control input Eq. (23) to the unforced equation can be designed to simultaneously drive the system to the target state and negate the undesirable transient effect of the control inputs to the forced equation. This can be done by modifying Eqs. (21) and (22) to be:

$$g(x) = \int_0^x \left(z_t^*(\sigma) - z_{pt}(\sigma, T)\right) d\sigma + (z^*(x) - z_p(x, T)), \tag{24}$$

$$h(x) = \int_0^x \left(z_t^*(\sigma) - z_{pt}(\sigma, T)\right) d\sigma - (z^*(x) - z_p(x, T)), \tag{25}$$

where $z_p(x, t)$ is the solution of the boundary value problem Eqs. (10)–(12) and $z_{pt}(x, t)$ is its time derivative.

Another role of the forced component is to encode the target equilibrium state through the function $f(x)$. The system equilibrium state $z_{eq}(x)$, obtained by setting the second time derivative to zero in Eq. (10), is the solution to the resulting boundary value problem,

$$c^2 \frac{d^2 z(x)}{dx^2} = -f(x), \quad z(0) = u_{e0}, \quad z(1) = u_{e1} \tag{26}$$

where u_{e0} and u_{e1} are the equilibrium values of $u_0(t)$ and $u_{1b}(t)$, respectively. When $f(x) = n^2\pi^2 \sin(n\pi x), n \in \mathbb{Z}^+$, and $u_{e0} = u_{e1}$, we see that $z_{eq}(x) = \frac{1}{c^2} \sin(n\pi x) + u_{e0}$. Since the sine series forms a complete basis of $L^2[0, 1]$, the set of square integrable functions over the domain [0 1], any function in $L^2[0, 1]$ can therefore be designed as a target state and reached at least in an average sense. For example, for the desired target state $z_d(x) = sin(2\pi x) + sin(6\pi x)$, we could assign $f(x) = c^2(2\pi)^2 sin(2\pi x) + c^2(6\pi x)^2 sin(6\pi x)$ and $u_{e0} = u_{e1} = 0$. In order to implement nonzero boundary conditions, u_{e0} and u_{e1} may be nonzero to shift the datum of the sinusoids from the origin.

5 Simulation Results

We validated our formation control approach for populations of $N = 100$ and $N = 1000$ agents. The agents start at the origin and are required to reach their final equilibrium configuration in time $T = 2$ s. The agent equilibrium configurations were specified along three target curves: a line, a circle, and a 3D closed curve in the form of a Lissajous knot.

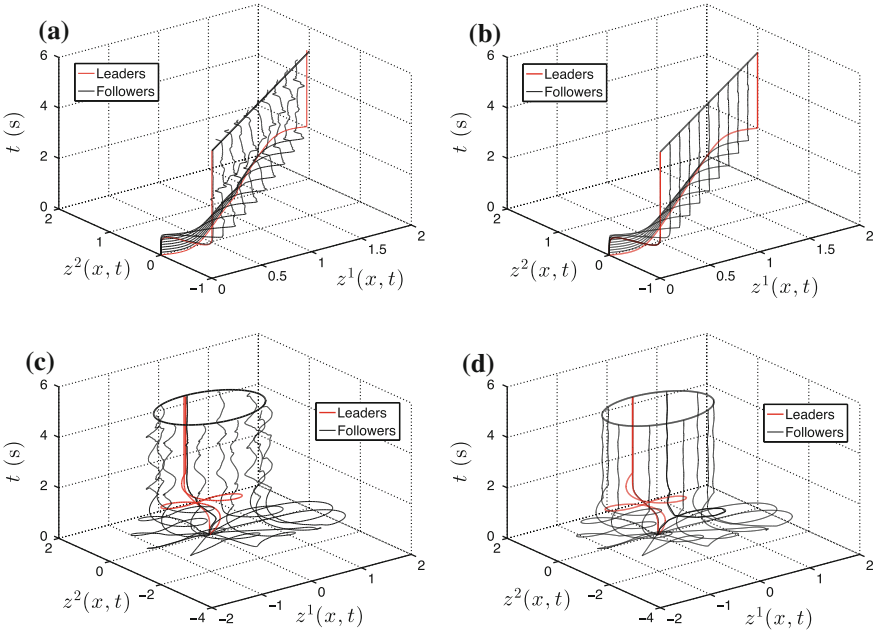


Fig. 1 Evolution of agent trajectories from the origin to a target line or a target circle at equilibrium. For clarity, the trajectories of only 10 agents are shown in each plot. **a** Line, $N = 100$ agents. **b** Line, $N = 1000$ agents. **c** Circle, $N = 100$ agents. **d** Circle, $N = 1000$ agents

Figures 1 and 2 show the time evolution of the agent positions for both population sizes and each type of target curve. The plots in Fig. 1 demonstrate that for populations of both $N = 100$ and $N = 1000$, the agent positions remain near the target curve when $t \geq T = 2\,\text{s}$. For both target curves, the population of 1000 agents exhibits smaller oscillations around the desired equilibrium positions than the population of 100 agents. This is because the network with the larger number of agents more closely approximates the continuum PDE model from which the control inputs are derived. The snapshots in Fig. 2 show that the networks of 100 and 1000 agents have similar transient dynamics (*i.e.*, similar agent position distributions at $t = 0.1\,\text{s}$ and $t = 0.5\,\text{s}$), but that by $T = 2\,\text{s}$, the larger network has converged much closer to the target 3D curve than the smaller network.

Figure 3 shows the time evolution of the absolute errors between the actual agent positions and their designed equilibrium positions along a circle for both $N = 100$ and $N = 1000$. The plots show that the agent position errors decrease markedly after $T = 2\,\text{s}$, indicating that the agent positions approach the desired equilibria in the required amount of time. The population of 1000 agents clearly displays smaller oscillations about these equilibria than the population of 100 agents.

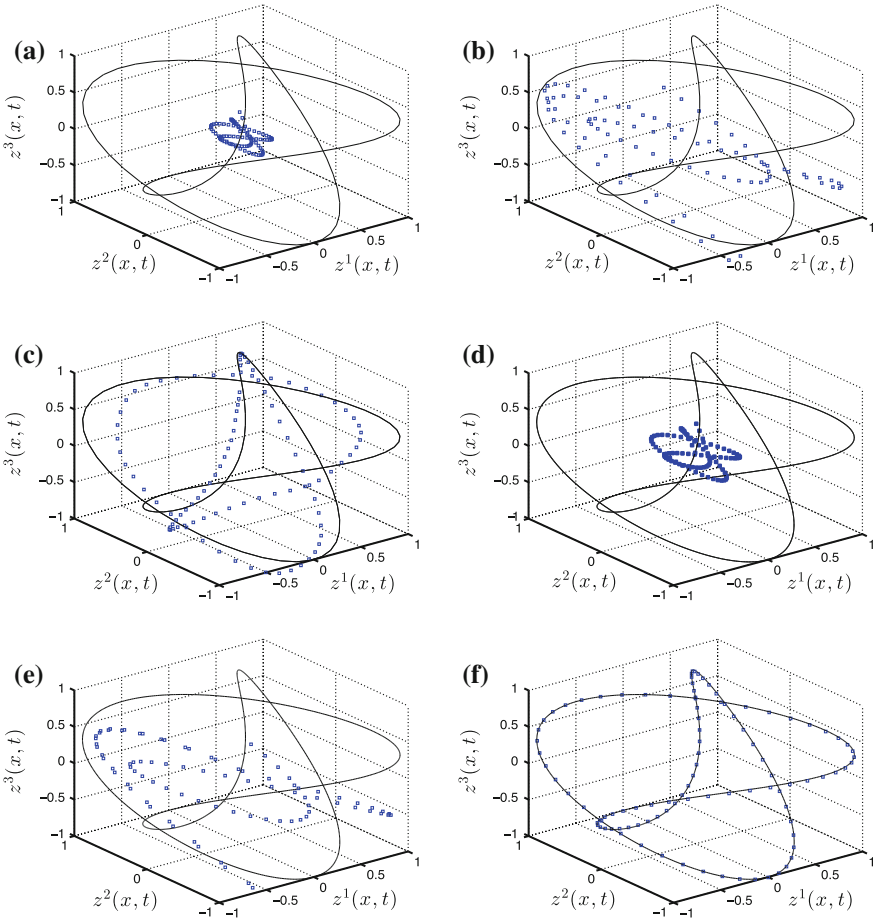


Fig. 2 Snapshots of agent positions (*blue markers*) at $t = 0.1$, 0.5, and 2 s as they converge from the origin to a target 3D curve (*black line*). The positions of 100 agents are shown in each plot. **a** $N = 100$, $t = 0.1$ s. **b** $N = 100$, $t = 0.5$ s. **c** $N = 100$, $t = 2$ s. **d** $N = 1000$, $t = 0.1$ s. **e** $N = 1000$, $t = 0.5$ s. **f** $N = 1000$, $t = 2$ s

6 Conclusions and Future Work

We have presented a trajectory planning methodology for a formation control problem on a chain network of agents with double integrator dynamics. Our approach is based on a wave equation abstraction of the system dynamics, and it is scalable with the number of agents and produces bounded control inputs. Despite the marginal stability of the system, the resulting open-loop control laws successfully drive the system to a target equilibrium state.

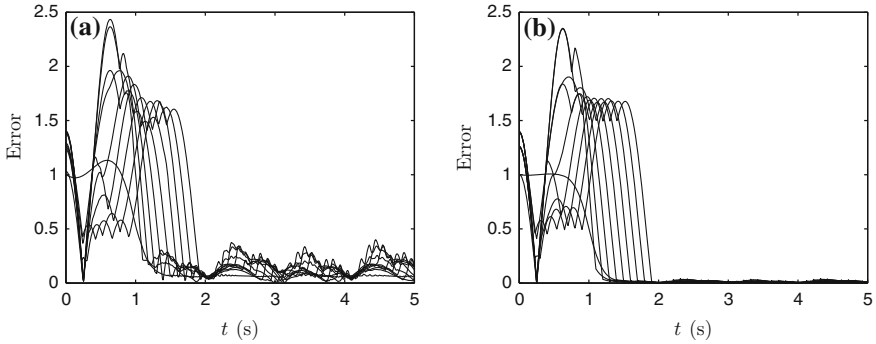


Fig. 3 Time evolution of the agent position errors when the target curve is the circle shown in Fig. 1c, d. For clarity, the position errors of only 10 agents are shown in both plots **a** Circle, $N = 100$ agents **b** Circle, $N = 1000$ agents

Due to the open-loop nature of the control strategy and the approximation error between the continuous and discrete models, our method shows higher accuracy in achieving the desired state as the number of agents in the network increases. Future work is needed on including feedback stabilization or other compensation schemes so that a wider class of systems is controllable using this approach.

It was also observed that double integrator networks have certain fundamental limitations for one-dimensional agent interconnection topologies. The wave equation has similar limitations in higher dimensions. Hence, graph topologies that have equivalent continuum approximations can be expected to have similar limitations. It would be interesting to see how the minimum time for controllability is reflected in double integrator networks with arbitrary topologies.

In addition, we plan to extend our methodology to multi-robot systems with realistic constraints and limitations. For instance, the robots' motion may be subject to holonomic constraints, which could be addressed by using infinite-dimensional equivalents of such networks with non-holonomic agents [19]. Robust control tools for distributed parameter systems [6] could be applied to systems with stochasticity and uncertainty in the robot dynamics. We will also need to account for possible loss of network connectivity and dynamically changing network topologies.

Furthermore, we would like to develop methods for formation control of systems that can be modeled by a wider class of linear and nonlinear PDEs. Additional variability can be incorporated either by using more leader agents or changing the control gains. For this reason, controlling a multi-agent system using nonlinear interconnection schemes can increase the set of reachable states at equilibrium. Another direction for future work is modeling grid networks of higher dimensions, and thus enabling deployment to formations on two-dimensional and three-dimensional manifolds.

References

1. Bamieh, B., Jovanovic, M.R., Mitra, P., Patterson, S.: Coherence in large-scale networks: dimension-dependent limitations of local feedback. IEEE Trans. Autom. Control **57**(9), 2235–2249 (2012)
2. Barooah, P., Mehta, P.G., Hespanha, J.P.: Mistuning-based control design to improve closed-loop stability margin of vehicular platoons. IEEE Trans. Autom. Control **54**(9), 2100–2113 (2009)
3. Bertozzi, A.L., Kemp, M., Marthaler, D.: Determining environmental boundaries: asynchronous communication and physical scales. In: Cooperative Control, pp. 25–42. Springer (2005)
4. Bliman, P.A., Ferrari-Trecate, G.: Average consensus problems in networks of agents with delayed communications. Automatica **44**(8), 1985–1995 (2008)
5. Chapman, A., Mesbahi, M.: Advection on graphs. In: Proceedings of the IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 1461–1466. IEEE (2011)
6. Curtain, R.F., Zwart, H.: An Introduction to Infinite-Dimensional Linear Systems Theory, vol. 21. Springer (1995)
7. Daganzo, C.F.: Requiem for second-order fluid approximations of traffic flow. Transp. Res. Part B: Methodol. **29**(4), 277–286 (1995)
8. Frihauf, P., Krstic, M.: Leader-enabled deployment onto planar curves: a PDE-based approach. IEEE Trans. Autom. Control **56**(8), 1791–1806 (2011)
9. Glowinski, R., Lions, J.L.: Exact and approximate controllability for distributed parameter systems. Acta Numer. **3**, 269–378 (1994). doi:10.1017/S0962492900002452
10. Jovanovic, M., Bamieh, B.: On the ill-posedness of certain vehicular platoon control problems. IEEE Trans. Autom. Control **50**(9), 1307–1321 (2005)
11. Leonard, N.E., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. In: Proceedings of the IEEE Conference on Decision and Control (CDC), vol. 3, pp. 2968–2973. IEEE (2001)
12. Meurer, T.: Control of Higher-Dimensional PDEs. Springer (2012)
13. Meurer, T., Krstic, M.: Finite-time multi-agent deployment: a nonlinear PDE motion planning approach. Automatica **47**(11), 2534–2542 (2011)
14. Micu, S.: Uniform boundary controllability of a semidiscrete 1-D wave equation with vanishing viscosity. SIAM J. Control Optim. **47**(6), 2857–2885 (2008)
15. Ogren, P., Egerstedt, M., Hu, X.: A control Lyapunov function approach to multi-agent coordination. In: Proceedings of the IEEE Conference on Decision and Control (CDC), vol. 2, pp. 1150–1155. IEEE (2001)
16. Pasqualetti, F., Zampieri, S., Bullo, F.: Controllability metrics, limitations and algorithms for complex networks. IEEE Trans. Control Netw. Syst. **1**(1), 40–52 (2014). doi:10.1109/TCNS.2014.2310254
17. Pimenta, L.C., Michael, N., Mesquita, R.C., Pereira, G.A., Kumar, V.: Control of swarms based on hydrodynamic models. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1948–1953. IEEE (2008)
18. Ren, W., Beard, R.: Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications. Springer (2007)
19. Rouchon, P.: Motion planning, equivalence, infinite dimensional systems. Int'l. J. Appl. Math. Comput. Sci. **11**, 165–188 (2001)
20. Sarlette, A., Sepulchre, R.: A PDE viewpoint on basic properties of coordination algorithms with symmetries. In: Proceedings of the IEEE Conference Decision and Control, held jointly with the Chinese Control Conference (CDC/CCC), pp. 5139–5144. IEEE (2009)
21. Sira-Ramirez, H., Agrawal, S.K.: Differentially Flat Systems, vol. 17. CRC Press (2004)
22. Woittennek, F.: On flatness and controllability of simple hyperbolic distributed parameter systems. In: Proceedings of the 18th IFAC World Congress, pp. 14,452–14,457. Milano, Italy (2011). doi:10.3182/20110828-6-IT-1002.02618

23. Yan, G., Ren, J., Lai, Y.C., Lai, C.H., Li, B.: Controlling complex networks: how much energy is needed? Phys. Rev. Lett. **108**(21), 218703 (2012)
24. Zuazua, E.: Propagation, observation, and control of waves approximated by finite difference methods. SIAM Rev. **47**(2), 197–243 (2005)

Decoupled Formal Synthesis for Almost Separable Systems with Temporal Logic Specifications

Scott C. Livingston and Pavithra Prabhakar

Abstract We consider the problem of synthesizing controllers automatically for distributed robots that are loosely coupled using a formal synthesis approach. Formal synthesis entails construction of game strategies for a discrete transition system such that the system under the strategy satisfies a specification, given for instance in linear temporal logic (LTL). The general problem of automated synthesis for distributed discrete transition systems suffers from state-space explosion because the combined state-space has size exponential in the number of subsystems. Motivated by multi-robot motion planning problems, we focus on distributed systems whose interaction is nearly decoupled, allowing the overall specification to be decomposed into specifications for individual subsystems and a specification about the joint system. We treat specifically reactive synthesis for the GR(1) fragment of LTL. Each robot is subject to a GR(1) formula, and a safety formula describes constraints on their interaction. We propose an approach wherein we synthesize strategies independently for each subsystem; then we patch the separate controllers around interaction regions such that the specification about the joint system is satisfied.

Keywords Reactive synthesis · LTL · Collision avoidance · Motion planning

1 Introduction

Formal synthesis has gained prominence over the last few years as a promising method to design correct-by-construction controllers [3]. Formal synthesis consists of a formal model of the object that is to be controlled and a formal specification of the property this object along with the synthesized controller need to satisfy. The problem has been studied for several classes of models including discrete systems,

S.C. Livingston (✉)
California Institute of Technology, Pasadena, USA
e-mail: slivingston@cds.caltech.edu

P. Prabhakar
IMDEA Software Institute, Madrid, Spain
e-mail: pavithra.prabhakar@imdea.org

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_26

timed systems and hybrid systems and several classes of specification languages including linear-time temporal logic (LTL), computation tree logic (CTL), μ-calculus as well as timed logics such as Metric Temporal Logic; e.g., [9, 10, 17].

In this paper, we focus on models that are distributed systems. Our motivation is the problem of synthesizing controllers for multi-robot systems wherein each robot has a task to achieve. For simplicity we consider the setting with two robots, but extension to a multi-robot setting is straightforward. The two robots can operate fairly independently except that they perform their tasks on a common workspace and hence might need to jointly satisfy certain constraints such as collision avoidance. We formalize this problem as a synthesis problem involving three specifications: one for each robot and one on the joint workspace.

A naive approach to deal with controller synthesis for distributed systems wherein the system is flattened to a single system using a product construction is inefficient because the size of the flattened system grows exponentially in the number of components. Thus a variety of robot architectures and control methods have been proposed that avoid this by various assumptions about communication among the agents, types of tasks, and allocation of responsibility [5, 8, 15, 16]. Relatively little prior work considers distributed robotics where tasks are formally specified using LTL. However, we remark that distributed computer systems (no continuous dynamics) is a well-known context for formal synthesis, though there are undecidability results for the general case [18]. Chen and collaborators address the problem of synchronization and task allocation by exploiting previous results for trace-closed languages [6]. Unlike the present work, they do not consider uncertainty in the environment. Ozay and collaborators present a methodology for decomposition of a specification, thus exploiting some symmetry present in the setting of multiple-target tracking in a network of actuated cameras [14]. Like the present work, they consider GR(1) formulae that can handle nondeterministic (uncertain) environments. However, unlike the present work, their decomposition is top-down in that a global specification is initially given, from which separate component specifications are manually constructed. While here we assume a discrete abstraction is given, there is prior work using sampling-based motion planning [20].

Our systems are not completely decoupled, but interact in some minimal way. Hence, we propose the following approach to deal with the state-space explosion. We synthesize the controllers independently for each robot so as to satisfy their corresponding specifications. The simultaneous execution of the strategies however might violate the specification on the joint workspace. We identify the states which violate the joint specification and patch the individual controllers in such a way that they satisfy their original specification and also satisfy the joint specification. More precisely, we identify a neighborhood around the joint property violation point and resynthesize a joint strategy for the two robots in this neighborhood satisfying the individual specification as well as the joint specification. We then decompose this joint strategy to obtain the patching strategy for individual robots. Note that patching involves solving a synthesis problem on the joint state-space of the two robots restricted to a small neighborhood of this space.

2 Preliminaries

We are concerned with the control of robots that satisfy a specification expressed in a fragment of linear temporal logic (LTL) known as GR(1). These robots operate in a shared workspace; their interaction is important and expressed formally as part of a joint specification. While most of our treatment concerns dynamics of discrete systems, such systems are practically obtained from continuous systems by a process of abstraction [1] (or [19] for a textbook introduction). The basic idea is that a finite transition system is bisimilar—in a precise sense—to a hybrid system if transitions between cells in a partition of the state space are achievable in one system if and only if they are achievable in the other. We omit further detail here because it suffices for our purposes to know that at least some of the discrete variables were obtained from a continuous dynamical system and thus admit a notion of distance.

A specification is an LTL formula that formally describes how the system should behave. It is written in terms of finitely-valued variables, some of which may be uncontrolled, like a disturbance. In this paper we use the GR(1) fragment because it has useful structure that we can exploit [4]. Our treatment of the syntax and semantics is brief and informal; an introduction can be found, e.g., in [2].

LTL is an extension to Boolean logic for describing sequences of events. Syntactically, a Boolean logic formula can contain operators $\vee$ "or" or $\neg$ "negation", along with derivative operators $\wedge$ "and", $\implies$ "implies", and $\iff$ "if and only if". These operators can be combined with the Boolean constants True and False and finitely-valued variables. Depending on its domain, a variable may appear in a subformula with inequality, e.g., $x < 5$. A Boolean formula evaluates to True or False for a particular assignment of the variables that appear in it.

A variety of operators concerning both future and past events have been introduced in LTL [7]. In this paper, we only make use of three: $\Box$ "always", $\Diamond$ "eventually", and $\bigcirc$ "next". An LTL formula is evaluated with respect to an infinite sequence of assignments to variables. Let f be a Boolean formula. Then $\Box f$ is True if and only if f is True at every time step. $\Diamond f$ is True if and only if f is True at some future time. $\bigcirc f$ is True if and only if f is True at the next time step.

Let $\mathscr{X}$ be a set of environment (or "uncontrolled input") variables, and let $\mathscr{Y}$ be a set of system (or "controlled output") variables. The sets of states, i.e., assignments from the domains, of these variables are denoted $\Sigma_{\mathscr{X}}$ and $\Sigma_{\mathscr{Y}}$, respectively. A GR(1) formula is of the form

$$\theta_{\text{env}} \wedge \Box\, \rho_{\text{env}} \wedge \left(\bigwedge_{j=0}^{m-1} \Box \Diamond\ \psi_j^{\text{env}} \right) \implies \theta_{\text{sys}} \wedge \Box\, \rho_{\text{sys}} \wedge \left(\bigwedge_{i=0}^{n-1} \Box \Diamond\ \psi_i^{\text{sys}} \right) \quad (1)$$

where the various subformulae are as follows. First notice the analogous form of both sides of the implication in (1), $\varphi^a \implies \varphi^g$; the left-side φ^a is commonly called

the "assumption," and the right-side φ^g the "guarantee." θ_{env} and $\psi_0^{\text{env}}, \ldots, \psi_{m-1}^{\text{env}}$ are Boolean formulae in terms of $\mathscr{X} \cup \mathscr{Y}$. θ_{env} is a condition that any initial state is assumed to satisfy. $\psi_0^{\text{env}}, \ldots, \psi_{m-1}^{\text{env}}$ are liveness conditions; the environment must set the variables in $\mathscr{X}$ infinitely often so as to satisfy these. ρ_{env} is a Boolean formula in terms of $\mathscr{X} \cup \mathscr{Y} \cup \mathscr{X}'$ that constrains from any particular state how the environment may move, i.e., set variables in $\mathscr{X}$ at the next time step (hence the primed notation $\mathscr{X}'$). The right-side is defined analogously, with the noticeable difference that ρ_{sys} is in terms of $\mathscr{X} \cup \mathscr{Y} \cup \mathscr{X}' \cup \mathscr{Y}'$, thus governing how the system may move from a particular state and given an anticipated environment move. The conditions $\psi_0^{\text{sys}}, \ldots, \psi_{n-1}^{\text{sys}}$ are also called "system goals."

There exist algorithms for the synthesis of finite-memory strategies realizing a given GR(1) formula [4]. In previous work, we proposed an annotation for strategies that facilitates online patching for coping with uncertainty [12, 13]. More precisely, suppose we are given a new specification φ' from modification of the original φ, e.g., due to online sensing necessitating updating an environmental model. Under certain conditions, the algorithm in [13] provides a way to locally change an original strategy to recover correctness with respect to φ'. The present paper builds on that method, and thus we summarize here the relevant results and notation. Unless stated otherwise, all GR(1) formulae in this paper are assumed to be realizable. Let φ be a GR(1) formula. A finite-memory strategy can be represented as an automaton $A = (V, \delta, L)$, where V is a finite set of nodes, $\delta \subset V \times \Sigma_{\mathscr{X}} \times V$ is a transition relation, and $L : V \to \Sigma_{\mathscr{X}} \times \Sigma_{\mathscr{Y}}$ labels nodes with states. $(u, e, v) \in \delta$ is also denoted $\delta(u, e) = v$. An automaton is said to be a *strategy automaton* for φ if its transition relation selects a valid next state from any state reachable in a play under φ. A is said to be *winning* if all plays resulting from its application satisfy φ.

Without loss of generality, winning strategy automata in this paper are assumed to be equipped with reach annotations, which are defined as follows. Denote the set of nonnegative integers by $\mathbb{Z}_+$. Given φ of the form (1), a state s is said to be a i-system goal if s satisfies ψ_i^{sys}. π_i is the mapping providing the ith component of elements of a Cartesian product. E.g., if $x = (x_1, x_2) \in \mathbb{R}^2$, then $\pi_1(x) = x_1$.

Definition 1 (*adapted from* [12]) A *reach annotation* on a strategy automaton $A = (V, \delta, L)$ for a GR(1) formula φ is a function $\text{RA} : V \to \{0, \ldots, n-1\} \times \mathbb{Z}_+$ that satisfies the following conditions. Given $p < q$, the numbers between p and q are $p+1, \ldots, q-1$, and if $q \leq p$, then the numbers between p and q are $p+1, \ldots, n-1, 0, \ldots, q-1$.

1. For each $v \in V$, $\pi_2 \circ \text{RA}(v) = 0$ if and only if $L(v)$ is a $\pi_1 \circ \text{RA}(v)$-system goal.
2. For each $v \in V$ and $u \in \text{Post}(v)$, if $\pi_2 \circ \text{RA}(v) \neq 0$, then $\pi_1 \circ \text{RA}(v) = \pi_1 \circ \text{RA}(u)$ and $\pi_2 \circ \text{RA}(v) \geq \pi_2 \circ \text{RA}(u)$.
3. For any path $\langle v_1, v_2, \ldots, v_K \rangle$ such that $\pi_2 \circ \text{RA}(v_1) = \cdots = \pi_2 \circ \text{RA}(v_K) > 0$, there exists an environment goal ψ_j^{env} such that for all $k \in \{1, \ldots, K\}$, $L(v_k)$ does not satisfy ψ_j^{env}.
4. For each $v \in V$ and $u \in \text{Post}(v)$, if $\pi_2 \circ \text{RA}(v) = 0$, then there exists a p such that for all r between $\pi_1 \circ \text{RA}(v)$ and p, $L(v)$ is a r-system goal, and $\pi_1 \circ \text{RA}(u) = p$.

In addition to other uses, reach annotation plays a crucial role in the patching algorithm of [13]. If a new reach annotation can be constructed after modifying a strategy automaton, then we immediately have that the new automaton is winning.

3 Problem Formulation

We are now ready to present the problem of formal synthesis for multiple robots addressed in this paper. Let φ_1 be a GR(1) formula (recall (1)) defined in terms of the environment variables $\mathscr{X}_1$ and the system variables $\mathscr{Y}_1$. Similarly for φ_2, $\mathscr{X}_2$, and $\mathscr{Y}_2$, where we require that the sets of system variables are disjoint, i.e., $\mathscr{Y}_1 \cap \mathscr{Y}_2 = \emptyset$. (It may be that $\mathscr{X}_1 \cap \mathscr{X}_2 \neq \emptyset$.) As for the single robot case of the previous section, a state is an assignment of values to variables. Unless indicated otherwise, a state assigns values to all of $\mathscr{X}_1 \cup \mathscr{X}_2 \cup \mathscr{Y}_1 \cup \mathscr{Y}_2$. Notice there is some redundancy in referring to states $\Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{X}_2}$ when $\mathscr{X}_1$ and $\mathscr{X}_2$ share variables. All variables are finitely valued.

We will be concerned with a joint specification. To that end, we need a way to compose φ_1 and φ_2. This is achieved by introducing an operator $\otimes$ over the set of GR(1) formulae. Making the "assumption" and "guarantee" components of the separate robot formulae explicit, write $\varphi_1 = \left(\varphi_1^a \implies \varphi_1^g\right)$ and $\varphi_2 = \left(\varphi_2^a \implies \varphi_2^g\right)$, and then define $\varphi_1 \otimes \varphi_2$ as

$$\left(\varphi_1^a \wedge \varphi_2^a\right) \implies \left(\varphi_1^g \wedge \varphi_2^g\right).$$

This is clearly a GR(1) formula, hence the operation $\otimes$ is closed over the set of GR(1) formulae.

Though the sets of discrete variables $\mathscr{Y}_1$ and $\mathscr{Y}_2$ are disjoint—and thus may be assigned independently by each respective robot—the robots perform their tasks in a shared workspace and thus may also need to meet a specification written in terms of both. This is achieved by introducing a Boolean formula $\varphi_{1,2}$ in terms of $\mathscr{Y}_1 \cup \mathscr{Y}_2$ and requiring that it is always satisfied. Finally, the target specification is

$$\varphi_1 \otimes \varphi_2 \otimes \left(\square\, \varphi_{1,2}\right). \tag{2}$$

in which we have omitted the "assumption" portion of GR(1) formula $\square\, \varphi_{1,2}$.

This problem could be solved by synthesizing in a product space to obtain a joint strategy that simultaneously selects actions for both agents. To avoid exponential increase in problem size entailed by such an approach, we propose to exploit the availability of an indicator function on states that provides a sufficient condition for satisfaction of $\varphi_{1,2}$. Concretely, suppose that $\mathscr{Y}_1$ and $\mathscr{Y}_2$ describe agents with identical dynamics that are operating in a shared workspace. Suppose further that

$\varphi_{1,2}$ describes states where the agents are dangerously close to each other. Because some discrete variables are abstractions of physical positions of the agents, we can construct from Euclidean distances a function

$$d : \Sigma_{\mathscr{Y}_1} \times \Sigma_{\mathscr{Y}_2} \to \{0, 1\} \tag{3}$$

such that

$$d(s^1, s^2) > 0 \implies \big(\varphi_{1,2} = \text{True}\big) \tag{4}$$

where s^1 and s^2 are states of the variables in $\mathscr{Y}_1$ and $\mathscr{Y}_2$, respectively. Intuitively, the safety condition $\varphi_{1,2}$ can only be violated when d is not positive. If the occurrence of d positive is uncommon, then we may be able to solve φ_1 and φ_2 independently and then correct parts of the strategies where interaction occurs, i.e., where extra effort is required to ensure $\square\,\varphi_{1,2}$.

4 Solution Approach

In this section the proposed solution is outlined, and detailed algorithms are presented in Sect. 5. Throughout this paper it is assumed that time is in discrete steps, and all agents are synchronized. More precisely, time evolves as it would in the product game, where at each step the environment first selects a move, and then, observing this, the system (i.e., the team of robots viewed as a single entity) selects a move corresponding to a simultaneous assignment of all variables in $\mathscr{Y}_1 \cup \mathscr{Y}_2$. Thus the next state is formed, and the process is repeated. This assumption allows us to avoid addressing what is otherwise a key issue in distributed robotics: synchronization. It is a reasonable approximation in the case of slowly moving robots, e.g., planar mobile robots, where such tools as the Network and Precision Time Protocols (IEEE standard 1588) are available. Furthermore, the specification (2) only requires the robots to be aware of each other to maintain the safety condition $\varphi_{1,2}$, which can be implemented passively using a range finder if this is collision avoidance. Finally, a shared environment (i.e., $\mathscr{X}_1 \cap \mathscr{X}_2 \neq \emptyset$) provides an external, event-triggered reference for coordination.

With the preceding assumption, synthesize winning strategies $A_1 = (V_1, \delta_1, L_1)$ and $A_2 = (V_2, \delta_2, L_2)$ for φ_1 and φ_2, respectively and independently, where φ_1 and φ_2 are as introduced in Sect. 3. Also compute reach annotations RA_1 and RA_2 for A_1 and A_2, respectively. Intuitively, we begin by treating the component specifications as being entirely separate and realize them using existing methods. To achieve (2), we must ensure that simultaneous execution of A_1 and A_2 does not lead to violation of $\varphi_{1,2}$. Since A_1 and A_2 were synthesized separately, it is possible that $\varphi_{1,2}$ is violated during their joint (simultaneous) execution. The proposed method addresses this in two parts:

1. During each time step, compute the set of nodes reachable by each robot's strategy A_i up to a horizon h. A dangerous configuration is one that may violate $\varphi_{1,2}$ and is checked for using the function (4) applied to the continuous positions of the robots. If no dangerous configurations are found within the designated horizon, then the robots' respective moves are performed. Else, proceed to the next part.
2. In order to guarantee avoidance of dangerous configurations, the robots must be aware of each other during motion planning. Accordingly, a local reachability game is constructed in terms of the joint specification (2). Its solution provides a local strategy for both robots to move from the current configuration, from which the danger was detected, to some configuration occurring after the danger. The notion of "after" is made precise by the reach annotations of A_1 and A_2, which are used to ensure that the respective goals in (2) are met infinitely often for the new robot strategies A_1' and A_2' that result from patching-in the local joint strategy.

5 Algorithms

The proposed method consists of two major steps: identification of dangerous configurations, and joint patching around those configurations. Our method is run online, and the main loop is listed in Algorithm 1. When dangerous configurations are found there, Algorithms 2 and 3 are invoked to patch the robots' control strategies. Comments on particular lines in those algorithms follow.

Algorithm 1 Main loop, including online detection of dangerous node pairs

INPUT: multi-robot task specification $\varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$, strategies A_1, A_2, reach annotations, $\mathrm{RA}_1, \mathrm{RA}_2$
Initialize with $(v_1, v_2) \in V_1 \times V_2$ depending on initial environment state.
while True do
  for k in $1, 2, \ldots, h$ **do**
    Compute $\mathrm{Post}_1^k(v_1)$ and $\mathrm{Post}_2^k(v_2)$.
    if $d(s^1, s^2) = 0$ for some $(s^1, s^2) \in L_1(\mathrm{Post}_1^k(v_1)) \times L_2(\mathrm{Post}_2^k(v_2))$ **then**
      $D_1 \times D_2 := \left\{(u^1, u^2) \in \mathrm{Post}_1^k(v_1) \times \mathrm{Post}_2^k(v_2) \mid d(L_1(u^1), L_2(u^2)) = 0\right\}$
      Invoke Algorithm 2 with v_1, v_2 and the dangerous node pairs of A_1 and A_2.
      if Algorithm 2 returned $(A_1', \mathrm{RA}_1', A_2', \mathrm{RA}_2')$ **then**
        Replace A_1, RA_1 and A_2, RA_2 with the returned patched versions.
      else
        abort //Local reachability game unsolvable
      end if
    end if
  end for
  Each robot observes environment move: e_1, e_2
  $v_1 := \delta_1(v_1, e_1)$; $v_2 := \delta_2(v_2, e_2)$ //Take moves
end while

Algorithm 2 Jointly patch strategies

INPUT: joint GR(1) formula $\varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$, strategies A_1, A_2, reach annotations RA_1, RA_2, current node-pair (v_1, v_2), and respective danger nodes D_1, D_2
OUTPUT: new component strategy automata A_1', A_2', and new reach annotations RA_1', RA_2'
$i := \pi_1 \circ \mathrm{RA}_1(v_1)$ //Relevant goal mode for first robot
$j := \pi_1 \circ \mathrm{RA}_2(v_2)$ //Relevant goal mode for second robot
$m_i := \min_{d \in D_1} \pi_2 \circ \mathrm{RA}_1(d)$ //Min. reach annotation among dangerous nodes for first robot
$m_j := \min_{d \in D_2} \pi_2 \circ \mathrm{RA}_2(d)$
$B_1 := \{v \in V_1 \mid \pi_1 \circ \mathrm{RA}_1(v) = i \wedge \pi_2 \circ \mathrm{RA}_1(v) < m_i\}$
$B_2 := \left\{v \in V_2 \mid \pi_1 \circ \mathrm{RA}_2(v) = j \wedge \pi_2 \circ \mathrm{RA}_2(v) < m_j\right\}$
Entry $:= \{(v_1, v_2)\}$
Exit $:= B_1 \times B_2$
$A_{(i,j)} := \mathrm{Reach}_\varphi(L(\mathrm{Entry}), L(\mathrm{Exit}))$
if $A_{(i,j)} = \mathtt{nil}$ **then**
  abort //Local reachability game unsolvable
end if
$(A^i, A^j) := \mathrm{Decompose}(A_{(i,j)})$
Patch A_1 with A^i and A_2 with A^j as in [13].
return $A_1', \mathrm{RA}_1', A_2', \mathrm{RA}_2'$

Algorithm 3 Decompose local strategy

INPUT: $A_{(i,j)} = \left(V_{(i,j)}, \delta_{(i,j)}, L_{(i,j)}\right)$ with state labels in $\Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1} \times \Sigma_{\mathscr{X}_2} \times \Sigma_{\mathscr{Y}_2}$
OUTPUT: A^i with state labels in $\Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1}$, and A^j with state labels in $\Sigma_{\mathscr{X}_2} \times \Sigma_{\mathscr{Y}_2}$
$V^i := \emptyset$; $V^j := \emptyset$
for $(v_1, v_2) \in V_{(i,j)}$ **do**
  $V^i := V^i \cup \{v_1\}$; $V^j := V^j \cup \{v_2\}$
  for $(u_1, u_2) \in \mathrm{Pre}((v_1, v_2))$ **do**
    $\delta^i(u_1, e_1) := v_1$, where $e_1 = L_1(v_1) \downarrow \Sigma_{\mathscr{X}_1}$
    $\delta^j(u_2, e_2) := v_2$, where $e_2 = L_2(v_2) \downarrow \Sigma_{\mathscr{X}_2}$
    if $L_1(u_1) = L_1(v_1)$ **then**
      $L^i(v_1) := L_1(v_1) \oplus \mathrm{Hash}(v_1)$
    else
      $L^i(v_1) := L_1(v_1)$
    end if
    if $L_2(u_2) = L_2(v_2)$ **then**
      $L^j(v_2) := L_2(v_2) \oplus \mathrm{Hash}(v_2)$
    else
      $L^j(v_2) := L_2(v_2)$
    end if
  end for
end for
return $A^i = \left(V^i, \delta^i, L^i\right)$, $A^j = \left(V^j, \delta^j, L^j\right)$

5.1 Comments on Algorithm 1

- Line 3 : because the specification describes infinite plays by the robots, the main loop should run forever to provide for online usage.

- Line 4 : to ensure that a dangerous configuration is not stepped over during the search, horizon lengths must be in increasing order, as listed in the for-loop of Algorithm 1.
- Line 5 : $\text{Post}_1^k(v_1)$ is the set of nodes in A_1 reachable in k time steps beginning at v_1, for some sequence of environment moves. $\text{Post}_2^k(v_2)$ is defined similarly for the second robot. For clarity, Algorithm 1 does not include the obvious improvement of incrementally computing $\text{Post}_2^k(v_2)$ using the value from the previous iteration of the for-loop.
- Line 6 : the predicate is in terms of continuous robot states; recall (4).
- Line 7 : compute all "dangerous" nodes that satisfy the predicate of line 6. Those of D_1 belong to the first robot; those of D_2 to the second.
- Line 16 : recall that there is only one environment, though each robot may see different parts of it. Their respective perspectives are indicated by using e_1 and e_2. On the next line, these are used to move to the appropriate next strategy automaton nodes.

5.2 Comments on Algorithm 2

- Lines 5–6 : for each robot strategy, compute the minimum reach annotation value for all nodes that are part of a dangerous configuration.
- Lines 7–8 : for each robot strategy, find all existing automaton nodes with the current goal mode and that have reach annotation strictly less than all dangerous nodes.
- Line 11 : L is the product labeling constructed from L_1 and L_2. Using the transition rules and safety conditions of the joint specification $\varphi := \varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$, solve a reachability game that drives any play initially from a state in $L(\text{Entry})$ to some state in $L(\text{Exit})$, or else block one of the environment liveness conditions (recall (1)) if this is not possible. For brevity we omit a code segment for constructing a strategy automaton realizing a solution. Many algorithms to do this are known, for instance a μ-calculus fixed point is used in [13]. A reachability game is defined as an LTL formula, called $\text{Reach}_\varphi(L(\text{Entry}), L(\text{Exit}))$,

$$\chi_{L(\text{Entry})} \wedge \square\,\rho_{\text{env}} \wedge \left(\bigwedge_{j=0}^{m-1} \square\diamond\ \psi_j^{\text{env}} \right) \implies \square\,\rho_{\text{sys}} \wedge \diamond\ \chi_{L(\text{Exit})}$$

  for which a strategy must be synthesized. The solution may be found by restricting attention to a set of states within a distance of the dangerous configuration, as described in [13]. While we omit details here, the basic idea is to form a smaller synthesis problem by modifying ρ_{env} and ρ_{sys} given the subset of states over which patching is performed.
- Line 13 : local reachability games can be unsolvable even when the multi-robot specification (2) has solutions. In the present context, there are two possible causes.

First, if one of the Entry will inevitably lead to violation of the inter-robot safety requirement $\varphi_{1,2}$. Second, we only treat the case of patching within a particular goal mode. Accordingly, it is not necessary to consider intermediate satisfaction robots' liveness goals as part of solving local reachability games. However, this may result in a trivially unsolvable reachability game, e.g., if B_1 is empty. Consult discussion in Sect. 6 concerning extension to the general case.

- Line 15 : invoke subroutine Decompose(), as provided in Algorithm 3, for decomposing local strategies into corresponding local strategies for separate agents, on respective discrete states $\Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1}$ and $\Sigma_{\mathscr{X}_2} \times \Sigma_{\mathscr{Y}_2}$.
- Line 16 : having obtained local strategies A^i and A^j on the previous line, which may be used to patch A_1 and A_2 for goal modes i and j, respectively, the single-agent algorithm of [13] can now be applied directly. We omit it for brevity.

5.3 Comments on Algorithm 3

- Lines 7–8: the notation $e_1 = L_1(v_1) \downarrow \Sigma_{\mathscr{X}_1}$ indicates projection onto the set of environment states from the perspective of agent 1, i.e., $\Sigma_{\mathscr{X}_1}$. Thus, the transition $\delta^i(u_1, e_1) := v_1$ means that upon reaching node u_1 in any play, if the environment takes the move e_1, then the strategy automaton will select a system state in $\Sigma_{\mathscr{Y}_1}$ such that the resulting discrete state in $\Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1}$ is $L(v_1)$.
- Lines 9–18 : avoid stuttering in component local strategies. We assume a unique number generator Hash() is available and append it to the state labelings on lines 10 and 15.

6 Analysis

As shown below, the proposed method is sound in the sense that infinite executions by a team of robots using it will realize the specification.

Theorem 2 *Any infinite play resulting from the combined operation of Algorithms 1–3 is correct with respect to specification $\varphi_1 \otimes \varphi_2 \otimes \Box\varphi_{1,2}$, provided the initial state does not violate $\varphi_{1,2}$.*

Proof Let σ be an infinite play, i.e., a mapping

$$\sigma : \mathbb{N} \to \Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1} \times \Sigma_{\mathscr{X}_2} \times \Sigma_{\mathscr{Y}_2}$$

that assigns to each discrete time step a product state. Denote the projection onto the variables used by the first and second robots respectively by

$$\sigma_1 : \mathbb{N} \to \Sigma_{\mathscr{X}_1} \times \Sigma_{\mathscr{Y}_1}, \tag{5}$$

$$\sigma_2 : \mathbb{N} \to \Sigma_{\mathscr{X}_2} \times \Sigma_{\mathscr{Y}_2}, \tag{6}$$

so that $\sigma(t) = (\sigma_1(t), \sigma_2(t))$ for each time t. The proof proceeds by induction. By hypothesis, the first state $\sigma(1)$ must not violate $\varphi_{1,2}$. Since A_1 and A_2 were synthesized for φ_1 and φ_2, respectively, the initial states of both robots $\sigma_1(1)$ and $\sigma_2(1)$ satisfy the initial conditions of φ_1 and φ_2, respectively. Therefore the joint initial state $\sigma(1)$ satisfies the initial conditions of the specification $\varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$.

Suppose that for some time t, the finite play fragment $\sigma(1) \cdots \sigma(t)$ satisfies the safety requirements of the specification. In terms of the components of the specification, satisfaction of the safety requirements means that for each $\tau \leq t$, $\sigma(\tau)$ satisfies $\varphi_{1,2}$, and for each transition $(\sigma(\tau), \sigma(\tau+1))$, the transition requirements of φ_1 are met by $(\sigma_1(\tau), \sigma_1(\tau+1))$ and the transition requirements of φ_2 are met by $(\sigma_2(\tau), \sigma_2(\tau+1))$ for $\tau \leq t-1$. The robots' strategy automata A_1 and A_2 transition on line 17 of Algorithm 1, which from the labelings determines the next state, i.e.,

$$\sigma_1(t) = L_1(v_1) \tag{7}$$

$$\sigma_2(t) = L_2(v_2) \tag{8}$$

$$\sigma_1(t+1) = L_1(\delta_1(v_1, e_1)) \tag{9}$$

$$\sigma_2(t+1) = L_2(\delta_2(v_2, e_2)). \tag{10}$$

Observe that L_1 in (7) may differ from L_1 in (9), since A_1 may have changed as a result of patching (Algorithm 2). A similar statement applies to L_2 in (8) and (10). Thus we consider two cases. For the first case, if patching does not occur, i.e., if the condition of the if-statement on line 6 of Algorithm 1 is always false, then the robot strategy automata are not changed in the present iteration. In particular, the labelings L_1 and L_2 are not changed and it suffices to check that the transitions resulting from δ_1 and δ_2 are safe. If these transitions are as in the originally given A_1 and A_2, then by hypothesis they are safe with respect to φ_1 and φ_2, respectively. They are also safe with respect to $\varphi_{1,2}$ since the condition of the if-statement on line 6 was false in the present case. Summarizing the second case, the solution of the local reachability game in Alogrithm 2 implies that transitions in the modified A_1' and A_2' are safe with respect to the multi-robot specification. Therefore, by induction the safety requirements of the specification $\varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$ are always met.

It remains to show that the liveness requirements of $\varphi_1 \otimes \varphi_2 \otimes \square\varphi_{1,2}$ are satisfied. By the definition of GR(1) formulae (1), this occurs if one of the environment liveness conditions is eventually never satisfied, or if all robots' goals are repeatedly achieved. Note that $\square\varphi_{1,2}$ is not relevant here, and therefore, it suffices to check the first robot goals as in φ_1 and the second robot goals as in φ_2. By hypothesis, we have an infinite play and thus the abort-statement in Algorithm 2 must never occur, i.e., for every time that patching is attempted, a solution is found for the local reachability game. That means a product strategy automaton $A_{(i,j)}$ ("product" in the sense of concerning both robots' system variables $\mathscr{Y}_1$ and $\mathscr{Y}_2$) is found in which either an environment liveness condition is blocked or strict progress is made toward the respective robots' goals (modes i and j as appearing in φ_1 and φ_2, respectively). Because Algorithm 3

decomposes this joint solution of the local reachability game into strategy automata A^i and A^j for each robot so that simultaneous execution of A^i and A^j is identical to $A_{(i,j)}$, it follows that the results A'_1 and A'_2 of patching robot strategy automata ensure the robot goals are repeatedly reached, or else a liveness condition in φ_1 or φ_2 is blocked.

An important part of the hypothesis for the previous theorem is that it concerns infinite plays. However, during usage of the presented algorithms, a patching attempt may fail and result in a finite play. To simplify the presentation, the algorithms in this paper rely on patching to be completed within the same goal mode. While an extension to the method of [13] has been developed that can visit robot goals while connecting Entry and Exit sets, it has not yet been published and length constraints prevent inclusion of it here. Thus our present treatment is restrictive and may fail to find a joint strategy when one exists.

7 Experiments

In this section preliminary results concerning the complexity of the method presented in this paper are described. The experiment setting considered is illustrated in Fig. 1. The underlying dynamics can be driven among cells using gradient methods [11], so to simplify the presentation we model the robots as entirely discrete transition systems.

Informally, the multi-robot specification in Fig. 1 requires that both robots visit both stars repeatedly while avoiding collisions with each other. It can be shown that, in the absence of an adversarial environment, winning strategies are of the form of "lassos," with a prefix and suffix loop; example paths are illustrated by dashed lines in the figure. In this example, a collision state occurs at row 5, column 7, and it would first be found h-steps away, where h is the maximum horizon parameter used in Algorithm 1.

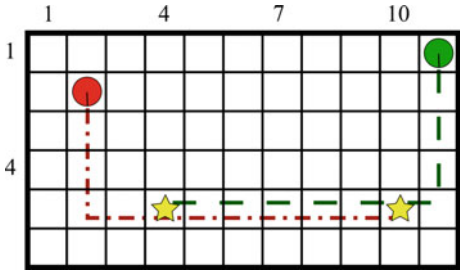


Fig. 1 Illustration of experiment setting: two-robot gridworlds. Cells that are to be visited repeatedly are indicated by *stars*. The robots are shown in their initial positions

7.1 *Quantifying Coupling*

The usefulness of the proposed method depends largely on how "loose" is the coupling of the robots imposed by $\varphi_{1,2}$ (recall (2)), given that nominal strategy automata constructed for φ_1 and φ_2 assume independence. We studied this in the gridworld setting as follows. Generate a random gridworld with fixed static obstacle density 0.2 (i.e., 20 % of cells are occupied), and create two robot specifications φ_1 and φ_2 in it by randomly placing, for each robot, one initial position and multiple goal positions. Synthesize strategy automata A_1 and A_2 independently for φ_1 and φ_2, respectively. Compute the synchronous product of A_1 and A_2 (viewed as finite transition systems), and find all nodes that are labeled with the same grid position, i.e., all nodes at which the robots would be in collision. These are referred to as "dangerous configurations", and histograms of occurrences for varying numbers of goals and world sizes is shown in Fig. 2. It is clear that in all cases, most counts are zero, i.e., the independently synthesized robot strategies A_1 and A_2 never result in dangerous configurations. While this observation holds as well for both gridworld sizes considered, the plots indicate

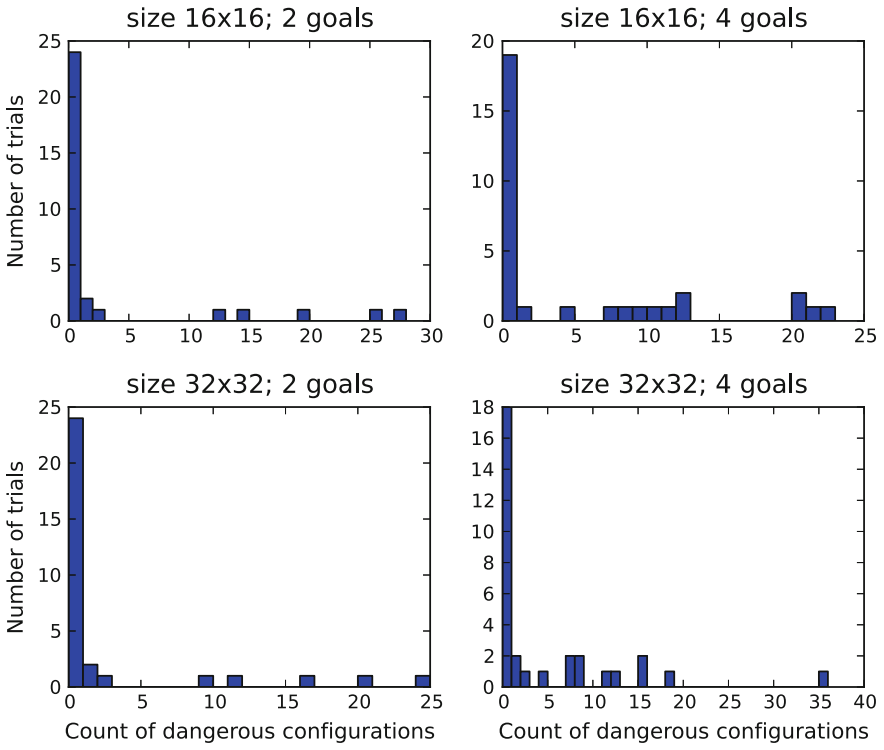


Fig. 2 Histograms for the number of trials in which a given count of dangerous configurations occurred, after generating multi-robot specifications for random gridworld instances as described in Sect. 7.1. In each case 32 trials were performed

Table 1 Automaton synthesis times for multi-robot random gridworlds

Setting	Mean time (s) for specification $\varphi_1 \otimes \varphi_2$	Mean time (s) for distributed synthesis
Size 16×16, 2 goals	0.938	0.196
Size 16×16, 4 goals	1.75	0.297

that increasing the number of goal positions increases the occurrences of dangerous configurations. Intuitively we may expect this because the goal positions are randomly placed in the gridworld and thus may cause motion plans to cover more of the workspace, thereby increasing possibilities for collisions.

7.2 Simulation Trials

Using the experiment setting described in the previous section, the times required for independently synthesizing strategy automata are compared with those required for using a combined specification. In terms of the previous section, strategy automata are synthesized for φ_1, φ_2, and $\varphi_1 \otimes \varphi_2$. Synthesis times are shown in Table 1.

References

1. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proc. IEEE **88**(7), 971–984 (2000)
2. Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press (2008)
3. Belta, C., Bicchi, A., Egerstedt, M., Frazzoli, E., Klavins, E., Pappas, G.J.: Symbolic planning and control of robot motion: finding the missing pieces of current methods and ideas. IEEE Robot. Autom. Mag. 61–70 (2007)
4. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**, 911–938 (2012)
5. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Applied Mathematics Series. Princeton University Press (2009). http://coordinationbook.info
6. Chen, Y., Ding, X.C., Stefanescu, A., Belta, C.: Formal approach to the deployment of distributed robotic teams. IEEE Trans. Robot. **28**(1), 158–171 (2012)
7. Emerson, E.A.: Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics, chapter Temporal and Modal Logic, pp. 995–1072. MIT Press (1990)
8. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
9. Karaman, S., Frazzoli, E.: Vehicle routing problem with metric temporal logic specifications. In: Proceedings of the 47th IEEE Conference on Decision and Control (CDC), pp. 3953–3958, Cancun, Mexico, December (2008)
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications. In: Proceedings of the American Control Conference (ACC), pp. 735–742, Montréal, Canada, June (2012)

11. Lindemann, S.R., LaValle, S.M.: Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions. Int. J. Robot. Res. **28**(5), 600–621 (2009)
12. Livingston, S.C., Murray, R.M.: Hot-swapping robot task goals in reactive formal synthesis. Technical report, California Institute of Technology, September (2014). http://resolver.caltech.edu/CaltechCDSTR:2014.001
13. Livingston, S.C., Prabhakar, P., Jose, A.B., Murray, R.M.: Patching task-level robot controllers based on a local μ-calculus formula. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 4573–4580, Karlsruhe, Germany, May (2013)
14. Ozay, N., Topcu, U., Wongpiromsarn, T., Murray, R.M.: Distributed synthesis of control protocols for smart camera networks. In: International Conference on Cyber-physical Systems (2011)
15. Parker, L.E.: Current state of the art in distributed autonomous mobile robotics. In: Proceedings of Distributed Autonomous Robotic Systems, vol. 4, pp. 3–12. Springer, Japan (2000)
16. Parker, L.E.: Distributed intelligence: overview of the field and its application in multi-robot systems. J. Phys. Agents **2**(1), 5–14 (2008)
17. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '89, pp. 179–190, New York, USA. ACM (1989)
18. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, October (1990)
19. Tabuada, P.: Verification and Control of Hybrid Systems: A Symbolic Approach. Springer (2009)
20. Zhu, M., Otte, M., Chaudhari, P., Frazzoli, E.: Game theoretic controller synthesis for multi-robot motion planning, Part I: trajectory based algorithms. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1646–1651, Hong Kong, China (2014)

Part IV
Multi-Robot Communication and Control Architecture

Knowledge Co-creation Framework: Novel Transfer Learning Method in Heterogeneous Multi-agent Systems

Hitoshi Kono, Yuta Murata, Akiya Kamimura, Kohji Tomita and Tsuyoshi Suzuki

Abstract This paper presents a framework, called the knowledge co-creation framework (KCF), for the heterogeneous multi-robot transfer learning method with utilization of cloud-computing resources. A multi-agent robot system (MARS) that utilizes reinforcement learning and transfer learning methods has recently been deployed in real-world situations. In MARS, autonomous agents obtain behavior autonomously through multi-agent reinforcement learning and the transfer learning method enables the reuse of the knowledge of other robots' behavior, such as for cooperative behavior. These methods, however, have not been fully and systematically discussed. To address this, KCF leverages the transfer learning method and cloud-computing resources. In prior research, we developed a hierarchical transfer learning (HTL) method as the core technology of knowledge co-creation and investigated its effectiveness in a dynamic multi-agent environment. The HTL method hierarchically abstracts obtained knowledge by ontological methods. Here, we evaluate the effectiveness of HTL with two types of ontology: action and state.

Keywords Transfer learning · Reinforcement learning · Heterogeneous agents · Multi-agent system

H. Kono (✉) · Y. Murata · T. Suzuki
Tokyo Denki University, Tokyo, Japan
e-mail: hitoshi@nrl.c.dendai.ac.jp

Y. Murata
e-mail: y.murata@nrl.c.dendai.ac.jp

T. Suzuki
e-mail: tszk@ieee.org

A. Kamimura · K. Tomita
National Institute of Advanced Industrial Science and Technology (AIST),
Ibaraki, Japan
e-mail: kamimura.a@aist.go.jp

K. Tomita
e-mail: k.tomita@aist.go.jp

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_27

1 Introduction

Actual multi-agent robot systems (MARSs) that use reinforcement learning have recently been deployed in real-world situations. Among other applications, a multi-robot inspection system for disaster-stricken areas, autonomous multi-robot security systems, and autonomous multi-robot conveyance systems for warehouses have been developed [1–3]. However, the real world, where such MARSs are expected to operate, is a dynamic environment that complicates the development of the systems because developers must customize the robots to this dynamic environment. The application of multi-agent reinforcement learning (MARL) to MARSs is one of the approaches taken in response to this problem. MARL is a mechanism for implementing a posteriori cooperation among agents, which can behave adaptively in a dynamic environment even when they are not provided with specific control policies. The benefits of MARL have been demonstrated in various studies over the past decade [4–6]. The application of MARL to actual robots has been studied by Matarić [7]. A method for accelerating the learning process has also been investigated because reinforcement learning in dynamic environments requires a long time to obtain an optimal (or nearly optimal) solution [6]. However, this method is difficult to apply to MARS with MARL in dynamic environments because the learning speed is impractically low. Moreover, a MARS typically contains at least one pre-programmed robot, and MARL has the following drawbacks.

- The learning process requires a long time.
- The obtained knowledge depends on the situation.
- There is a limit to a robot's capacity to store the knowledge.

In contrast, cloud robotics has recently been proposed [8, 9] as a means to increase the availability of standalone robots by utilizing cloud computing resources. Cloud robotics may increase the utility of MARSs because the robots gain access to broader knowledge, vast computing resources, and external functions. This should be helpful for achieving practical implementation of MARSs with MARL.

In this context, we propose a knowledge co-creation framework (KCF) by in-tegrating MARS, MARL, and cloud robotics [10, 11]. We developed KCF by leveraging the transfer learning method and cloud-computing resources. To implement this framework, an autonomous mobile robot in a MARS internally executes cyclical processes, and we implement cloud services for gathering and assimilating knowledge (Fig. 1) as follows.

- Knowledge data are generated by using computer simulation and other MARL systems.
- A robot saves knowledge to its own repository via a network connected to cloud computing resources.
- The robot observes the environmental state.
- The robot selects particular knowledge from the repository on the basis of the observed environment.

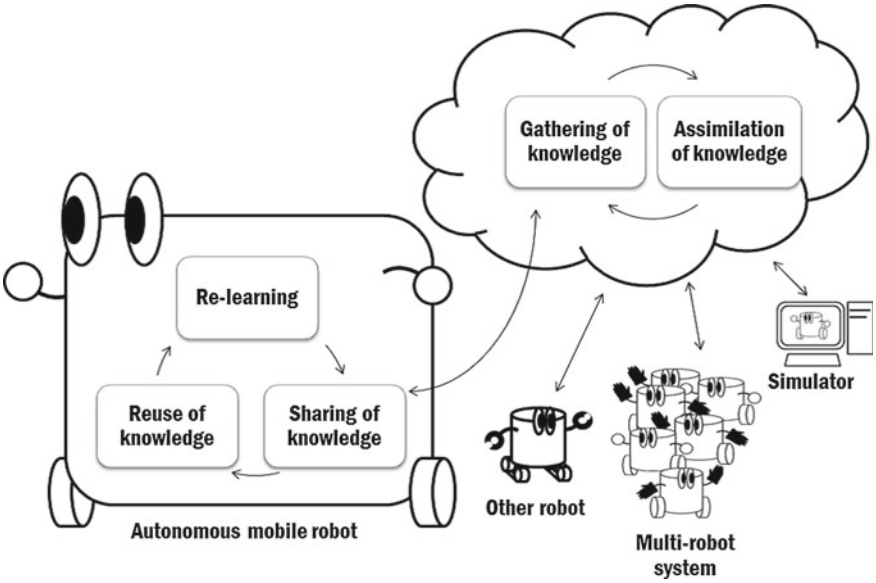


Fig. 1 Simplified representation of KCF. All systems (including the MARS simulator) are connected to cloud-computing resources

- If the observed environment is unknown, the robot acquires the learned knowledge of other robots (reuse of knowledge) [12].
- As a result of this action, the robot obtains new knowledge about unknown en-vironments and shares new knowledge with other robots and systems.

Note that an autonomous agent acts on the basis of existing knowledge if the ob-served environment is known.

We developed the hierarchical transfer learning (HTL) method as the core technology of KCF. The HTL method enables inter-task mapping (ITM) by using action ontology among heterogeneous agents. This allows autonomous robots and virtual agents to reuse knowledge from other types of agents. Here, we describe experiments that confirm that HTL enables reuse of knowledge by using action and state ontologies to mediate between heterogeneous MARSs.

The rest of the paper is organized as follows. Section 2 describes the theory and assumptions of reinforcement learning and transfer learning. Section 3 is an overview of the proposed KCF. Section 4 provides details about the preconditions of simulation experiments. Section 5 details evaluation of the effectiveness of KCF through simulation and contains a discussion of the results, which suggest that autonomous learning agents can reuse knowledge from other heterogeneous agents by using KCF. Section 6 contains concluding remarks.

2 Learning Mechanisms and Transfer Method of Knowledge

2.1 Reinforcement Learning

Reinforcement learning is one type of machine learning, in which agents can use a trial-and-error method to create policies for accomplishing tasks. In this study, we adopt Q-learning, defined below, as the reinforcement learning mechanism:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma V(s') - Q(s,a)\} \tag{1}$$

Here, a value function $V(\cdot)$ is defined as follows.

$$V(s) = \max_{a \in A} Q(s,a) \tag{2}$$

Here, S is a state space, with $s, s' \in S$; a an element of an action space A; $\alpha (0 < \alpha \leq 1)$ is the learning rate; $\gamma (0 < \gamma \leq 1)$ is the discount rate; and r is the reward. Q-learning is simple algorithm, and it contributes straightforward implementation with programming language. The learning agents select each defined a with a probability given by the Boltzmann distribution according to

$$p(a|s) = \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A} \exp\left(\frac{Q(s,b)}{T}\right)} \tag{3}$$

Here, T is a parameter that determines the randomness of selection. The Q-learning model can select actions in descending order according to the action value from learned knowledge. When the values of available actions are the same or are equal to default value, the Boltzmann distribution is used to select the action at random.

2.2 Transfer Learning in Reinforcement Learning

Transfer learning, as proposed by Taylor, is a framework for reuse of a policy obtained through reinforcement learning [12]. The policies and solutions obtained through reinforcement learning are here regarded as knowledge. In the transfer learning method, an agent first learns the policy as an action-state pair during the source task. Next, an agent performing the target task can reuse the knowledge obtained during the source task via ITM. ITM defines the relation of the spaces S and A between the target and source tasks. If the target task agent has state space S_{target} and action space A_{target}, and the source task agent has state space S_{source} and action space

A_{source}, then ITM for simple tasks will map S and A between the target and source tasks. This is formulated as follows:

$$\chi_S(s) : S_{target} \rightarrow S_{source}$$
$$\chi_A(a) : A_{target} \rightarrow A_{source} \quad (4)$$

The agent completing the target task can have different characteristics from the agent that learned the source task. Hence, the agent performing the target task can adapt its behavior for a new environment or target task. This method is fundamental in a single-agent environment.

2.3 Transfer Learning in Multi-agent Domains

In recent years, transfer learning has been investigated not only for single-agent systems but also for MARSs. For example, Boutsioukis et al. proposed a transfer learning method with multi-agent reinforcement learning, which enables the use of ITM among agents [13].

Taylor et al. proposed a parallel transfer learning method, which runs the target and source tasks simultaneously [14]. Their method speeds up learning in multi agent transfer learning. However, many such methods do not take into account the operation of large numbers of single-agent systems and MARSs, which means that an inter-task map must be either created or modified with every entry of a new agent system.

The quality of ITM is the most important factor in agent performance on target tasks. Therefore, we believe that ITM for a system should be designed by humans (such as researchers and engineers) on the basis of experience and intuition. However, as already mentioned, manually designing an ITM system is problematic when large numbers of single-agent systems and MARSs are involved in the transfer learning system.

3 Hierarchical Transfer Learning

3.1 Ontology-Based ITMs

Our KCF with HTL enables integration of ITMs among agents [11]. In a previous paper, we proposed HTL, which uses the concept of ontologies as a method for creating ITMs. We call this technique ontology-based ITM (OITM). Ontology is introduced here as an "explicit specification of a conceptualization" for the purpose of learning [15]. Our OITM leverages functions by which we can describe many different relations in terms of ontology, and specifically we can describe integrative

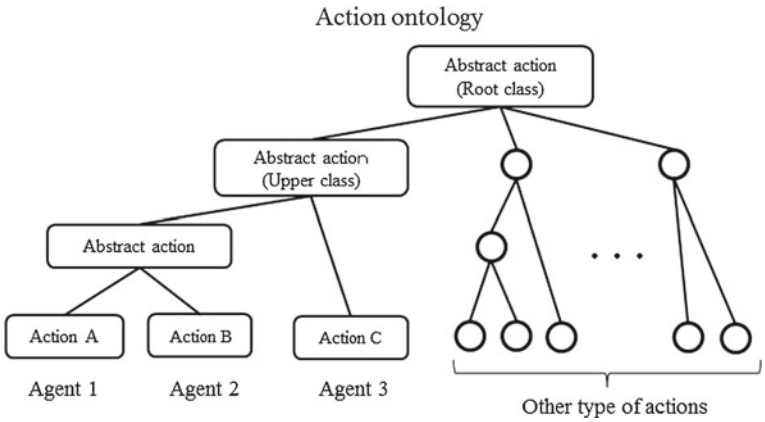


Fig. 2 OITM for agent actions. The agent's developer maps concrete actions of the agent to abstract actions as upper classes, and these upper classes can be mapped into higher classes. Therefore, all actions of all agents are mapped to an ontology

ITM among agents. Moreover, if we first define ITM of a system in terms of ontology, then agents can use ITM to search the knowledge of many other agents. We assume that a concrete action of an agent is called an instance of ontology and an abstract action of ontology is called a class or upper class. We additionally assume that any ontology present in cloud resources can be accessed by all agents.

An example of OITM is shown in Fig. 2. First, the agent developer maps the concrete action of an agent to the ontology. Another agent developer also maps an action to this ontology. When the agent reuses the knowledge of the other agent, it searches the mapping between its actions and other agent actions. Second, the agent transfers knowledge from other agents to itself using the knowledge and mapping of ontology for ITM. Note that when the agent transfers the knowledge from other agents, OITM requires two ontologies, such as an action ontology and a state ontology. Hence, the agent individually searches corresponding actions and states of other agents.

3.2 Transfer of Knowledge

As mentioned above, the agent can reuse knowledge of other agents through HTL. In this study, we adopted Q-learning as the reinforcement learning model. For the Q-learning mechanisms, transferred knowledge is reused as follows.

$$Q^c(s,a) = Q^t(s,a) + \tau Q^s(\chi_S^O(s), \chi_A^O(a)) \tag{5}$$

Here, $Q^t(s,a)$ is knowledge about the target task and $Q^s(s,a)$ is knowledge about a source task, via HTL. The transferred knowledge also uses OITM, and the function

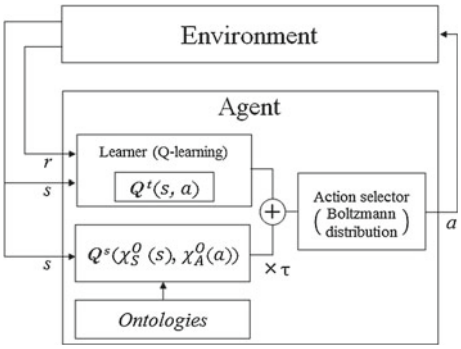


Fig. 3 Simplified internal reinforcement learning model in target task agent. A learner can receive state and reward from the environment, and transferred knowledge cannot receive the reward

of $\chi_S^O(\cdot)$ and $\chi_A^O(\cdot)$ means OITM. The term $Q^c(s, a)$ is the combined knowledge of the target and source tasks, and $\tau (0 < \tau \leq 1)$ is a parameter for adjusting the action's value for the difference between the target and source task. A target task agent selects an action from $Q^c(s, a)$ according to a Boltzmann distribution (Eq. (3)). However, updating of knowledge occurs only for $Q^t(s, a)$ by Q-learning (Fig. 3). In an actual environment, when an actual agent, such as a robot, reuses transferred knowledge, the knowledge of source tasks consists of a data file generated by the source task agent, and the target task agent must receive the transferred knowledge (in the form of these files) about the source task via the network infrastructure. Hence, to reuse knowledge, HTL requires a communication infrastructure, list of available repositories of knowledge and public ontology servers.

4 Task Description

4.1 Pursuit Game

Previous studies have adopted tasks such as zero-sum games, foraging tasks, and cooperative carrying tasks for evaluating MARL. Here, we adopt a pursuit game to evaluate performance. The pursuit game is a benchmark test of agent performance, measured as time to capture. We set an $N \times N$ grid as the world. An arbitrary number of hunter agents and prey agents are deployed in this world, and we evaluate the number of steps (i.e., time) until the hunters capture all prey. This game is over when the prey is captured, i.e., all hunters are adjacent to the prey at the end of a turn. In our pursuit game, the locations of all agents are reset to their initial positions after capture. A single episode is defined as the time until a state of capture has been reached. Agents act according to a predefined order, such as hunter 1 → hunter 2 → prey, and one set of actions is regarded as a single step. A cell cannot be simultaneously occupied by multiple agents, and agents cannot cross the world boundaries. Moreover, hunters can learn cooperative capture actions, but prey cannot learn.

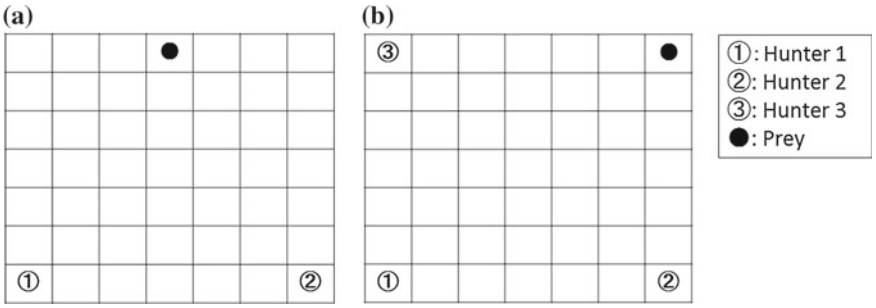


Fig. 4 Difference in tasks. **a** Two hunters versus one prey in 7 × 7 grid world, with initial positions of each agent. **b** Three hunters versus one prey in 7 × 7 grid world with initial positions of agents in the four corners

4.2 Difference in Tasks

We define the grid world of a pursuit game according to a study by Tan [4]. In this particular implementation, hunters and prey can move in a 7 × 7 grid world. The initial position of each agent is shown in Fig. 4. The difference between tasks is the number of hunters (Fig. 4a, b). We call the task in Fig. 4a "2 vs. 1" and that in Fig. 4b "3 vs. 1". Note that in the 2 versus 1 task, the observable environmental state of a hunter is the set containing the coordinates of the other hunter and of the prey. In the 3 versus 1 task, the observable environmental state is set containing the coordinates of the two other hunters and of prey. Therefore, the concrete difference between tasks is the observable number s of the set of S. In each task, the observable environmental state as a set S is defined as follows.

$$S_{2\,vs.\,1} = \{self\text{-}location,\ coordinates\ of\ other\ hunter,\ coordinate\ of\ prey\} \quad (6)$$

$$S_{3\,vs.\,1} = \{coordinates\ of\ self\text{-}location,\ coordinates\ of\ a\ second\ hunter,\ coordinates\ of\ a\ third\ hunter,\ coordinates\ of\ prey\} \quad (7)$$

4.3 Heterogeneity of Agents

As mentioned above, the game involves two types of agents: hunters and prey. Only hunters are provided with learning mechanisms; the actions of the prey are according to a fixed strategy, as discussed in detail below.

Agents can select only one action per step. Prey can choose from among five actions in an action space A_{prey}, which is defined as follows.

$$A_{prey} = \{front,\ back,\ right,\ left,\ stop\} \quad (8)$$

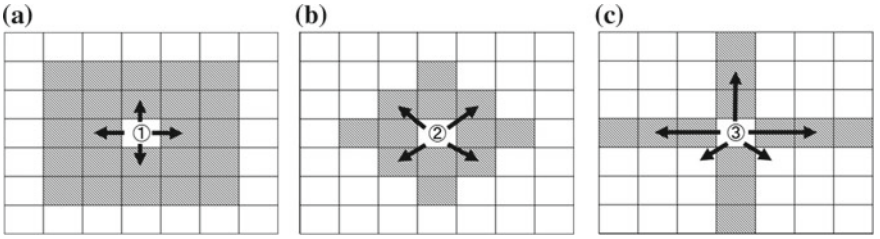


Fig. 5 Actions and sight range of each agent. *Arrows* denote movable direction and distance in grid world. *Gray* area is sight range of each agent, and if other agents are in sight range, agent can observe coordinate of other agents

Heterogeneity of hunters means that differences are permitted between the strategies and action spaces of different hunters. In addition, each agent is provided with a sensor, such as sight. We define the allowed actions of each hunter as follows.

$$A_{hunter1} = \{front,\ back,\ right,\ left,\ stop\} \tag{9}$$

$$A_{hunter2} = \{upper\ right,\ lower\ right,\ lower\ left, upper\ left,\ stop\} \tag{10}$$

$$A_{hunter3} = \{front\ (2\ \text{cell}),\ right\ (2\ \text{cell}),\ lower\ right, lower\ left, left\ (2\ \text{cell}), stop\} \tag{11}$$

Here, characteristics of $A_{hunter1}$, $A_{hunter2}$ and $A_{hunter3}$ is shown in Fig. 5a–c, respectively. Each agent has its own sight range (shown as shaded cells), and the shape of this range differs among agents. The sight range of the prey is the same as that shown in Fig. 5c. Initially, hunters and prey choose their actions randomly. Hunters adjust the probabilities with which actions are selected as the learning progresses. Although the prey does not learn, it selects an escape action when it recognizes a hunter. The prey moves away from the hunter when it detects only one hunter, or in any of the possible escape directions (uniformly chosen) when it detects multiple hunters in its vicinity.

4.4 Experimental Conditions

To confirm the effectiveness of transfer with HTL, we set the experimental conditions as listed in Table 1. In this experiment, we adopted the 2 versus 1 task as the source task, and the 3 versus 1 task as the target task. In the source task, hunters 1 and 2 and the prey are deployed in a 7×7 grid world. In the target task, hunters 1, 2, and 3 and the prey are deployed in a 7×7 grid world. If hunter 1 in the target task uses

Table 1 Experimental conditions of transfer

Experiment	Self-transfer			HTL		
	Source task		Target task	Source task		Target task
Task	2 versus1		2 versus 1	2 versus 1		3 versus 1
Hunters	(1) and (2)		(1) and (2)	(1) and (2)		(1), (2) and (3)
Direction of transfer	(1)	→	(1)	(1)	→	(1)
	(2)	→	(2)	(2)	→	(2)
				(1)	→	(3)

The target task agent uses transferred knowledge from a source task agent of the same type . Only hunter 3 of the target task does not have an analogous agent in the source task, and it can select the transferred knowledge of hunter 1. In this table (1) is Hutner1, (2) is Hunter2, and (3) is Hunter3

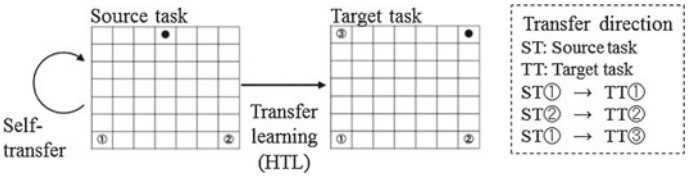


Fig. 6 Transfer learning and self-transfer. Transfer learning condition transfers knowledge from a source task to a target task. The self-transfer condition transfers knowledge from a source task to an identical task

transferred knowledge, it is transferred from hunter 1 in the source task. Hunter 2 in the target task also uses knowledge from hunter 2 of the source task. However, hunter 3 of the target task does not have an analogous agent in the source task. Under this experimental condition, hunter 3 can select the transferred knowledge of hunter 1 because hunter 3 is more similar to hunter 1 than to hunter 2 (Fig. 6). Moreover, on top of the above experimental conditions, we test the self-transfer condition. Self-transfer is used as confirmation of transferred knowledge properly generated by the agent of the source task, and we transfer the generated knowledge from the source task to the source task agents.

In the source task, the Q-learning parameters are set to $\alpha = 0.7$, $\gamma = 0.9$, and $r = 1$. The Boltzmann parameter T is 0.01. These parameters are common to the self-transfer condition, and the transfer rate τ is 1.0. In the target task, Q-learning parameters are set to $\alpha = 0.1$, $\gamma = 0.99$, and $r = 1$, and T is set to 0.01. The default Q-values are 0 in all experiments, and τ is 1.0. In each experiment, 5000 episodes are conducted for the source and target tasks, and we execute 10 trial.

When the agent has learned behavior in the source task, the obtained knowledge is formatted as a function approximation by using an artificial neural network (ANN) because the format of the obtained knowledge depends on the internal mechanism of the agent in the source task. An example of function approximation is shown in Fig. 7. Therefore, in the HTL framework, we adopted function approximations as

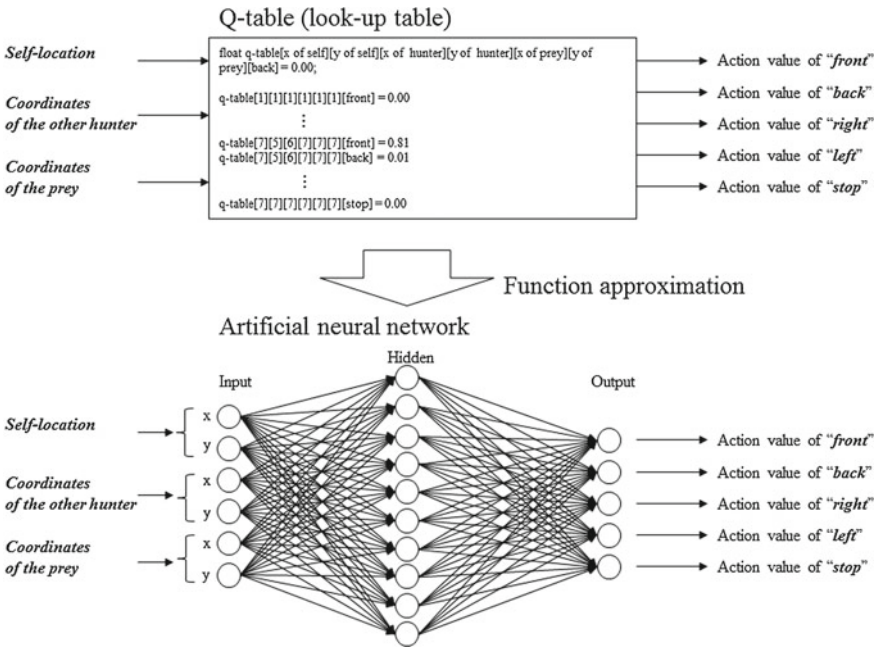


Fig. 7 Function approximation of obtained knowledge as Q-table. This table typically has action values for all of states. Its size becomes dramatically enlarged when the number of states is large. In our proposed method, we approximate the Q-table with artificial neural networks

the common format of knowledge among agents. Furthermore, parameters of ANN are set as follows. The number of input nodes = $S_{2vs.1}$. The number of hidden nodes is double number of input nodes and the number of output nodes is the number of elements of $A_{hunter1}$ or $A_{hunter2}$. We adopt a backpropagation algorithm for training ANN. Note that actual transferred data as obtained knowledge is ANN parameters such as the connection weights, the number of neurons and connection architecture.

In this experiment, we designed two ontologies for HTL; action ontology in Fig. 8 and state ontology in Fig. 9. For example, when the hunter 3 reuses the knowledge of hunter 1 by utilizing action ontology and state ontology, information of observed states are put in state ontology. The hunter 3 can translate own observed states to observable states of hunter 1, and translated states are input to the knowledge transferred from hunter 1. Then knowledge outputs the action values of hunter1, and hunter 3 translates it to own actions by utilizing action ontology. Finally, hunter 3 calculates joint knowledge (Eq. (5)), and it selects valuable action using Boltzmann distribution (Eq. (3)). Here we assume that the two ontologies are pre-programmed in the all hunters.

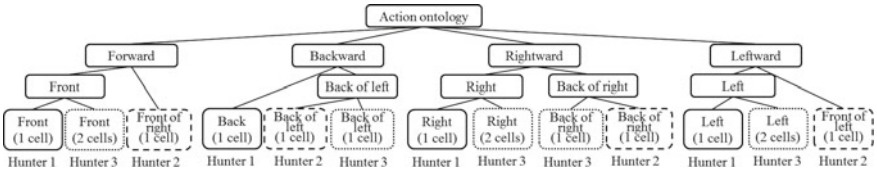


Fig. 8 Action ontology. In action ontology, we map the instance of actions to the similar upper class

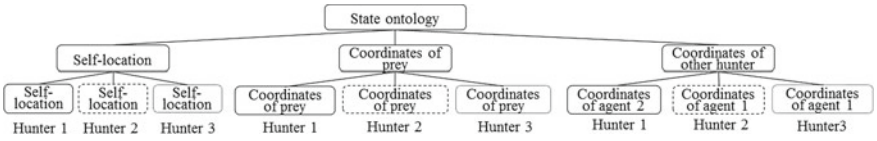


Fig. 9 State ontology. This ontology is according to the transfer from source task to target task

5 Experimental Results and Discussion

In this section, we describe the experimental results and discuss *Jumpstart* (JS), which is the mean difference in initial performance (average steps in initial 100 episodes) between an agent with transfer and one without transfer.

5.1 Self-transfer Results

In this experiment, the result of learning without transfer shows improved performance (Fig. 10). This learning curve does not converge to a single solution, in contrast to the performance of general reinforcement learning in a static environment, because the agents in all of our experiments learn in a dynamic environment. The learning curve of self-transfer exhibits an obvious JS compared with that without transfer. This is caused mainly by reuse knowledge via self-transfer. The average of improvement rate in JS is shown in Table 2. The average self-transfer of the initial 100 episodes improves 74 % from the source task without transfer. Moreover, the number of steps of the initial episode with transfer is low compared with the number of steps in the limit of the learning curve without transfer. The main cause of this is the function approximation by the ANN. This suggests that when there is transfer, transferred knowledge is optimized. This phenomena is the secondary cause of function approximation aimed at formatting of knowledge.

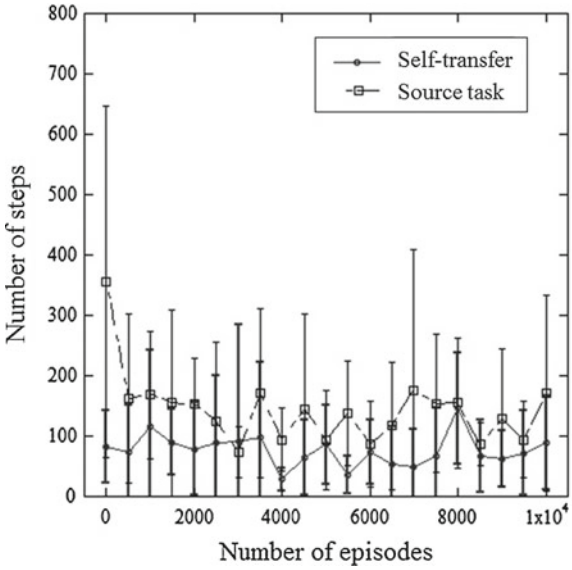


Fig. 10 Comparison of learning curves between source task and self-transfer (same domain target task). In this figure, each plot is average of 10 trials, and error bar means standard deviation

Table 2 Comparison of JS value, self-transfer, and HTL

Domain	Without transfer	With transfer	JS (improvement rate)
Source task (Self-transfer)	338.65	89.19	0.74
Target task (HTL)	4674.70	616.61	0.87

5.2 HTL Results

In HTL, the results exhibit an obvious JS, as shown in Fig. 11. The improvement rate in JS is shown in Table 2. These results indicate the effectiveness of HTL using OITM. The average of the initial 100 episodes scores with transfer is improved by 87 % compared with the average without transfer. This is caused mainly by reuse knowledge via HTL. As mentioned above, the number of steps of initial steps of learning curve with transfer is low compared with the steps in the limit of the system without transfer, and the main cause of this is also ANN. When the knowledge is approximated in source task, it is suspected that the action value of learned knowledge is optimized. The above results suggest that HTL can be used for transfer learning in heterogeneous MARSs.

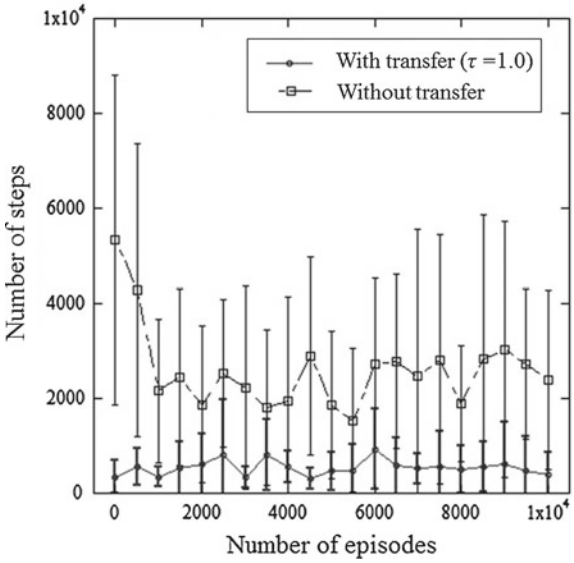


Fig. 11 Comparison of learning curves between with transfer and without transfer. The learning curve without transfer is normal reinforcement learning. These results show that the learning curve with transfer exhibits a JS compared with the learning curve without transfer. In this figure, each plot is average of 10 trials, and error bar means standard deviation

6 Conclusion

In this paper, we proposed KCF for implementation of MARL, and presented HTL as a transfer learning method suitable for large numbers of heterogeneous agents. The HTL method is one of the functions of KCF. Moreover, we also carried out an experiment under two transfer conditions by conducting computer simulation of a pursuit game. The experimental results suggest that HTL can transfer knowledge among heterogeneous agents and between different tasks. For future work, we plan to demonstrate the effectiveness of HTL by conducting experiments in actual multi-robot systems and more difficult tasks. In this experiment, the action sets and state sets were discrete, as was the experimental environment. As a continuation, HTL should be applied to continuous sets. A guideline for designing ontologies should be developed, which is important because the architectures of ontologies (e.g., instances and classes with relations among them) depend on the ontology developer's experience.

Acknowledgments This work was partially supported by Research Institute for Science and Technology of Tokyo Denki University Grant Number Q14J-01 Japan.

References

1. Sugiyama, H., Tsujioka, T., Murata, M.: Coordination of rescue robots for real-time exploration over disaster areas. In: The 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, Orlando, FL, pp. 170–177 (2008)

2. Marino, A., Parker, L.E., Antonelli, G., Caccavale, F.: A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. J. Intell. Robot. Syst. (2012). doi:10.1007/s10846-012-9783-5
3. d'Andrea, R.: Guest editorial: a revolution in the warehouse: a retrospective on Kiva systems and the grand challenges ahead. IEEE Trans. Autom. Sci. Eng. **9**(4), 638–639 (2012)
4. Tan, M.: Multi-agent reinforcement learning: independent vs. cooperative agents. In: 10th International Conference on Machine learning, pp. 330–337 (1993)
5. Arai, S., Sycara, K., Payne, T.R.: Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. In: PRICAI 2000 Topics in Artificial Intelligence, pp. 125–135 (2000)
6. Yang, E., Gu, D.: A survey on multiagent reinforcement learning towards multi-robot systems. In: IEEE Symposium on Computational Intelligence and Games, ID2012, Essex, UK (2005)
7. Matarić, M.: Reinforcement learning in the multi-robot domain. Auton. Robots **4**(1), 73–83 (1997)
8. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galván-López, D., Haussermann, K., Janssen, R., Montiel, J.M., Perzylo, A., Schießle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: A world wide web for robots roboearth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011)
9. Hu, G., Tay, W.P., Wen, Y.: Cloud robotics: architecture, challenges and applications. IEEE Netw. **26**(3), 21–28 (2012)
10. Kono, H., Sawai, K., Suzuki, T.: Convergence estimation utilizing fractal dimensional analysis for reinforcement learning. In: SICE Annual Conference 2013, pp. 2752–2757 (2013)
11. Kono, H., Kamimura, A., Tomita, K., Murata, Y., Suzuki, T.: Transfer learning method utilizing ontology for heterogeneous multi-agent reinforcement learning. Int. J. Adv. Comput. Sci. Appl. **5**(10), 156–164 (2014)
12. Taylor, M.E.: Transfer in Reinforcement Learning Domains. Studies in Computational Intelligence, vol. 216. Springer, New York (2009)
13. Boutsioukis, G., Partalas, I., Vlahavas, I.: Transfer Learning in Multi-agent Reinforcement Learning Domains. Recent Advances in Reinforcement, Learning, pp. 249–260 (2012)
14. Taylor, A., Dusparic, I., Galvn-Lpez, E., Clarke, S., Cahill, V.: Transfer learning in multi-agent systems through parallel transfer. In: 30th International Conference on Machine Learning (2013)
15. Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hbner, S.: Ontology-based integration of information—a survey of existing approaches. In: IJCAI-01 Workshop: Ontologies and Information Sharing, pp. 108-117 (2001)

Distributed Communication and Localization Algorithms for Homogeneous Robotic Swarm

Donghwa Jeong and Kiju Lee

Abstract Swarm robotics aims to achieve physical flexibility, overall system robustness, and enhanced reliability and efficiency by employing a group of autonomous robots for collective task performance. Achieving collective performance by individual robots with limited sensing, processing, and communication capabilities, however, faces several technical challenges, such as difficulties in establishing reliable communication and decentralized control among the robots. This paper presents the following wireless communication algorithms that can be applied to homogeneous swarm robots: (1) infrared-based short-distance communication between the adjacent robots using a self-synchronization technique; and (2) long-distance communication and localization based on distance measurement using radio signals. In addition, two decentralized global shape formation algorithms for homogeneous swarm robots are presented for simulating dispersion and line formation collectively achieved by homogeneous swarm robots.

Keywords Swarm robots · Wireless communication network · Global shape formation · Distributed sensing

1 Introduction

Social insects, such as ants, bees, and termites, exhibit collective behavior by sensing local information, communicating with each other, and sharing information [1]. For instance, ants carry food in an optimal path using a pheromone to create a chemical trail followed by other ants [2]. Bees and termites construct complex hives or caves in a decentralized manner without a global leader [1, 3]. Inspired by such biological swarm intelligence, robotics researchers have been developing distributed,

D. Jeong · K. Lee (✉)
Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland 44106, USA
e-mail: kiju.lee@case.edu
URL: http://case.edu/mae/robotics

D. Jeong
e-mail: donghwa.jeong@case.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_28

cooperative methods for a group of robots with limited functional capabilities to perform tasks that are typically difficult to be achieved by an individual robot. Several benefits are expected by using swarm robots over a single integrated system, including flexibility and accessibility due to the relatively small physical size, robustness achieved by the overall sustainability, and cost-effective control strategies by employing distributed control. For example, a couple of broken robots in a robotic swarm would not affect the overall task performance significantly. Despite these potential advantages, swarm robotics is also deemed to present several technical challenges in (a) forming wireless network using a specific type(s) of sensors; (b) establishing reliable and fast communication among the robots; and (c) controlling the flock of robots to perform a global task *without* centralized control.

In an attempt to address the above challenges, this paper presents the distributed sensing and communication algorithms in homogeneous networked robots with limited sensing, processing, and communication capabilities. We also present algorithms for global shape formation (i.e. dispersion and forming a line) to demonstrate global task performance via wireless communication.

Related Works

One of the primary goals of swarm robotics is to achieve collective task execution by communicating and collaborating with each other. To do so, the system must form a communication network among the swarm entities for detecting which robots are nearby and the distance between them, sending and receiving data among those within the communication range, and performing a task collectively. The specific task to be performed may vary depending on the application, but the reliable and efficient communication and localization capability serves as the fundamental functionality of the swarm system as a whole.

Ultrasound-, IR-, or RF-based technologies are most commonly used for wireless communication and localization in swarm robotics. Firstly, ultrasonic-based technology requires a pair of ultrasonic transmitter and receiver. For example, the Millibot system used a sonar sensor with an acoustic reflector for reflecting incoming acoustic signals toward the ultrasound transducer [4]. An ultrasonic relative positioning system with high accuracy of about ± 4 mm and a 3 m range was developed for localization in multi-robot systems [5, 6]. The experiments on this positioning system were conducted using only two robots and suffered from echo effects causing poor performance when more than two robots were employed in the test. Instead, communication and localization techniques using IR has several advantages over ultrasonic-based approaches. Although the ultrasonic-based technology is relatively cheap, the accuracy is lower in comparison with IR-based technology. Also, the whole infrastructure of the IR method is simpler than one for ultrasonic [7]. In addition, interference in ultrasonic-based communication is often higher than IR-based communication [8]. Alice and E-Puck are the robotic hardware platforms that used IR-based technologies for robot-to-robot communication [9, 10]. The Alice microrobot is driven by

two wheels for locomotion and is equipped with four IR modules for short-range communication with neighboring robots [9]. E-puck is a hockey puck-like circular robot driven by two stepper motors for locomotion. E-puck also has eight IR modules to achieve short-distance communication [10]. Although IR is still the most commonly used technology for wireless communication in swarm robotics, it has a significant drawback that IR signals can be easily blocked or interfered with by an object because of the high directivity.

The methods using radio signals attempt to overcome the limitations of the IR-based techniques. Radio signals have wider coverage than IR, while the hardware implementation is still easy and the data-communication link is robust [11–13]. Localization using the RF data is typically achieved by processing the Received Signal Strength Indicator (RSSI) values, converting this data into physical distances, and performing triangulation or trilateration for localization [14, 15]. To further enhance performance on finding the radio source, some studies employed additional hardware structures, such as antennae, to increase directivity [16, 17]. These studies showed improved direction sensing achieved by adding a relatively simple hardware structure(s), but neither study was fully validated for a large number of robots. Unfortunately, radio signals can be easily interfered with by obstacles or redirected by another antenna in close proximity. Moreover, fast and precise RSSI acquisition is difficult due to settling time for receiving the data packet. In addition, a small amount of data requires a large amount of data packets to be transmitted to the other receiver reliably. In theory, RSSI packets can be sent in less than a millisecond [18]. However, due to the existing problems such as latency, distance from the access point, and interference, actual transmission time is more than a millisecond. Particularly, collisions occur during bi-directional communication increases transmission time in high-speed modules as there is no mutual time coordination. A relevant study revealed that a transmitter can send only 170–180 packets in 30 s [19].

There exist several communication protocols for establishing a wireless network using radio or IR, including ZigBee, Bluetooth, Wi-Fi, NFC, and IrDA. Table 1 summarizes these protocols in terms of bandwidth, transmission range, power consumption, network size and type, cost, system complexity, and signal type. NFC (Near Field Communication) and Bluetooth are considered not suitable for swarm robotics due to limitations in network size. The Wi-Fi-based approach may be adopted for a small group of robots with high performance, but it is not desirable for swarm robotics due to the high system complexity and high cost. IrDA and ZigBee are widely accepted in swarm robotics because of the low complexity in hardware, relatively easy system implementation, and low power consumption. Despite the small bandwidths in these protocols, the swarm robots are expected to share only a limited amount of information through the wireless communication channel and therefore these protocols can accommodate the needs. Although the network size of IrDA is only 1, the module size is much smaller while the communication range is much larger than NFC. Therefore it is suitable for one-to-one communication among swarm robots.

Table 1 Comparison among different wireless communication protocols in terms of bandwidth, transmission range, power consumption, network size/type, cost, and system complexity

Properties	ZigBee	Bluetooth	Wi-Fi	NFC	IrDA
Bandwidth	20, 40, and 250 Kbit/s	<1.0 Mbit/s	11, 54 Mbit/s	424 Kbit/s	20–40, 115 Kbit/s, 4, 16 Mbit/s
Transmission range	<300 ft	<30 ft	<300 ft	<1 ft	<30 ft (line of sight)
Power consumption	Very low	Medium	High	Low	Low
Network size	32,000	1–7	<10	1	1
Network type	Ad-hoc	Ad-hoc	Point to hub	Point to point	Point to point
Cost	Low	Low	High	High	Low
System complexity	Low	Low	High	High	Low
Signal type	Radio	Radio	Radio	Radio	IR

In swarm robotics, like many other multi-robotic systems, synchronization in communication is one of the most challenging technical problems as it requires mutual time coordination among the multiple mobile nodes [20, 21]. It becomes particularly important for small and low-cost robots as they carry a limited amount of energy [21]. One of the commonly used approaches for synchronization involves a specific reference node generating a logical clock that can be shared among the multiple robots via wireless network such as radio frequency (RF) [21]. The reference node may be predesignated or it can be elected by a minimum number of connection nodes covering all the connected swarms [20]. As another approach, self-synchronization, or peer-to-peer synchronization without a master, can be realized using a variable-length source code [22]. However, a common problem with this approach is that channel errors may cause synchronization slippage.

2 Wireless Network for Communication and Localization

We consider a decentralized system consisting of a group of swarm robots where each robot has limited sensing capabilities and communication range. An IR-based method was adopted for synchronization via short-distance communication ($\leq$1 in.) and an RF-based approach was used for localization and long-distance communication ($\leq$60 in. for localization and 300 ft for communication). Detailed description on the developed communication and localization algorithms is provided in this section.

2.1 Hybrid Wireless Network Strategy Using IR and RF

Each IR- or RF-based method has its own advantages and disadvantages as discussed earlier. To establish a reliable and efficient wireless network among a group of swarm robots, we developed a hybrid algorithm that uses both IR and RF technologies. The IR-based method is adopted for precise short-distance communication within less than an inch for synchronization and the RF-based ZigBee technology is used for localization and long-distance communication as shown in Fig. 1. The general communication range of ZigBee is up to 300 ft where the range for localization by distance measurement using RSSI is limited to 60 in. Within this range, a linear relationship between the RSSI data and physical distances is observed [17]. For IR-based communication, we used pairs of an IR emitting diode and a phototransistor (QRD1114) instead of commonly used IR data association (IrDA) devices. These optical IR sensors are cheaper and easier to implement without requiring many additional hardware components than IrDA devices.

2.2 Self-synchronization via IR-Based Communication

Without a predetermined priority or a master-slave architecture, synchronization is required to initiate local communication between the robots. Synchronization is commonly performed by providing a common timescale for local clocks in the network. However, every hardware's clock is imperfect and may drift away from each other over time. Therefore, observed time or duration of time intervals may differ at each node in the network. Our algorithm uses an internal clock, transmitted pulses, and received pulses and compares these three for self-synchronization among the nodes (i.e. robots). When two robots get close to each other, the IR sensor in each robot transmits and receives pulses. By comparing these pulses and an internal clock, one of the robots decides to listen while the other talks. Ideally, for two nodes, **A** and **B**, **A** should listen while **B** talks, and vice versa in order to avoid any communication

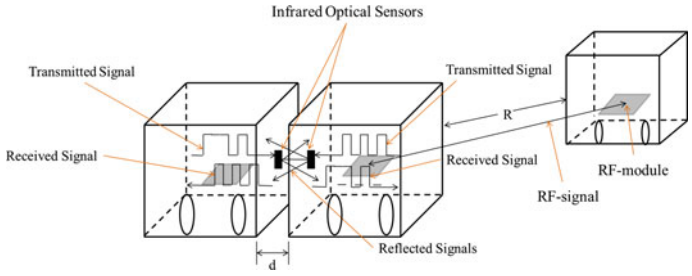


Fig. 1 Hybrid wireless network scheme: Short-distance communication using IR sensors and long-distance communication with RF modules: $d \leq 1$ in. and $R \leq 60$ in. The RF signal can reach up to 300 ft while reliable distance measurement for localization can be made within 60 in. of distance

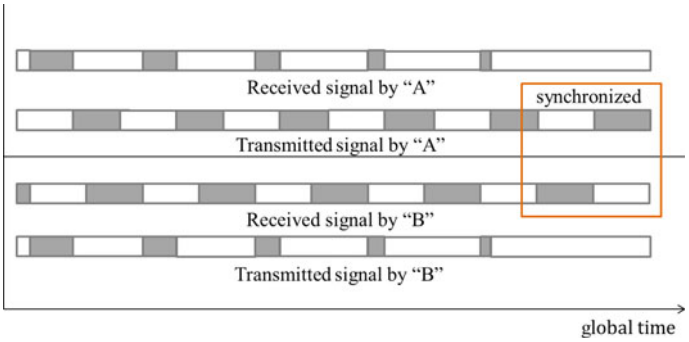


Fig. 2 Self-synchronization scheme with port *A* and port *B*; *gray bar* is talking bytes while the *white bar* indicates listening bytes

conflict and data loss. In general, however, the robots may not be fully synchronized and both may try to either talk or listen at the same time resulting in deadlocks. Because there is no priority in the communication order, shifting bits for synchronization in a certain direction is also not deterministic.

As illustrated in Fig. 2, once conflict in communication between **A** and **B** is observed, the talking bytes of **A** are shortened and synchronized after a couple of cycles. This bit-shift strategy is performed as follows, for the port **A**:

$$I_A = [1, 2, 3, \ldots, 14, 15, 16, 17, 18, 19, \ldots, 30, 31, 32]; \quad (1)$$

$$T_A = [0, 0, 0, \ldots, 0, 0, 0, 1, 1, 1, \ldots, 1, 1, 1]; \quad (2)$$

$$R_A = [*, *, *, \ldots, *, *, *, *, *, *, \ldots, *, *, *] \quad (3)$$

I_A is the 32-bits cyclic bit number index generated by the internal clock, T_A is the transmitted binary data, and R_A is the received signal from the other port. The pulse train from the T_A is generated from the IR photo diode according to the bit number of the I_A. When the two ports, **A** and **B**, are within the local communication range, R_A receives the data from T_B as well as reflected signal from T_A. Due to the pull-up resistors attached to the photo transistors, R_A is set as one initially or when no IR signals received and changes to zero when IR input is detected.

Algorithm 1 shows the pseudo-code for the self-synchronization technique. Shifting either forward or backward can be decided stochastically but can take a long duration. Every 31st bit of the 32 bits (4 bytes), the binary train moves one step forward or two steps forward based on the received data. The decision to shift either forwards or backwards for synchronization can be made stochastically; however, this process can take a long time. For every 31st bit of the 32 bits (4 bytes), the binary train moves one step forward or two steps forward based on the received data. If the $\sum_{n=bit1}^{bit8} R_A(n)$ is smaller than the $\sum_{n=bit9}^{bit16} R_A(n)$, then the pulse train proceeds one step forward whereas the pulse train moves two steps forward when $\sum_{n=bit1}^{bit8} R_A(n)$ is bigger than the $\sum_{n=bit9}^{bit16} R_A(n)$. Algorithm 1 is implemented in each node so in

Algorithm 1 Self Synchronization Technique for IR Sensors

while Not synchronized
if bit n = 31 **then**
  if $\sum_{bit1}^{bit8} R_A(\mathrm{n}) < \sum_{bit9}^{bit16} R_A(\mathrm{n})$ **then**
    bit n ← bit n+1
  else if $\sum_{bit1}^{bit8} R_A(\mathrm{n}) = \sum_{bit9}^{bit16} R_A(\mathrm{n})$ and $\sum_{bit1}^{bit16} R_A(n) = 16$ **then**
    bit n ← bit n+ r and (1,2)
  else if $\sum_{bit1}^{bit8} R_A(\mathrm{n}) = \sum_{bit9}^{bit16} R_A(\mathrm{n})$ and $\sum_{bit1}^{bit16} R_A(n) = 0$ **then**
    Synchronization is done!
  end if
end if
bit n ← bit n+1
if bit n = 33 **then**
  bit n ← bit 1
end if
end while

the case that **A** encounters $\sum_{bit1}^{bit8} R_A(\mathrm{n}) > \sum_{bit9}^{bit16} R_A(\mathrm{n})$, **B** will encounter $\sum_{bit1}^{bit8} R_B(\mathrm{n}) < \sum_{bit9}^{bit16} R_B(\mathrm{n})$. In this case, robot B would shift one bit forward and robot A would not shift. When the two transmitted signals are the same, a random shift is applied to each port so that it makes partial synchronization. When the synchronization is completed, IR communication data packets are replaced in the last two bytes of T_A. When the synchronization is completed, IR communication data packets are replaced in the last two bytes of T_X. Once the synchronization is completed, the following two bytes of T_X are replaced with the data packets (Fig. 3).

2.3 RF-Based Distance Measurement Using Bi-directional RSSI

To realize distance measurement in a decentralized manner, we developed a fast and reliable bi-directional RSSI technique to estimate distance and establish communication network in swarm robotics. All transceivers are programmed identically without a master. One common problem in bi-directional communication with high speed is that each transceiver may keep sending the data even when they collide with each other. The total time for sending and receiving signal, therefore, include the time for

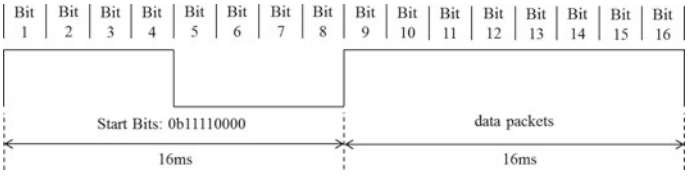


Fig. 3 IR communication data packets (two bytes of talking)

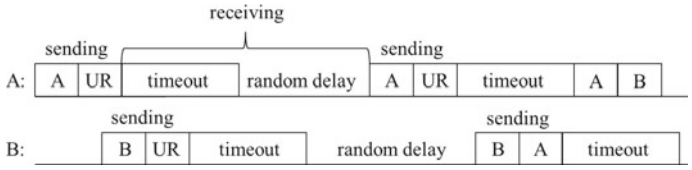


Fig. 4 ZigBee synchronization after adopting additional receiving period with random delay

CSMA-CA and retrials. To avoid this problem, a simple synchronization method is developed for the conflicted radio signals. Let **A** and **B** be arbitrary wireless transceivers without priority in between the two. Regardless of which one starts first, there will be a collision in communication (see Fig. 4). **A** sends packets consisting of two payloads: MY (my address), UR (your address). Initial UR is void when synchronization is not accomplished. During a fixed timeout period, each transceiver receives the packet and checks whether UR is void or filled with a received packet. If it is void, then the transceiver makes random time delay for receiving a packet and sending it afterwards. The collision-free bi-directional RSSI collection algorithm is described in Algorithm 2. The collected RSSI data can then be converted into physical distances using the log-distance path loss model and processed into position information via triangulation or trilateration [15].

Algorithm 2 Collision-free Bi-directional RSSI

```
1: while Not synchronized
2: Send packets
3: Receive packets for timeout
4: Read the first packet and save to packet[0]
5: Read the second packet and save to packet[1]
6: if packet[1] == MY then
7:     random delay = 0
8: else if packet[1] == 0 then
9:     random delay = random(0 to 50ms)
10: end if
11: end while
```

2.4 Algorithm Evaluation

The IR self-synchronization algorithm was evaluated by the synchronization time between two robots. The duration between the times when an IR port of a robot receives an IR signal from another port and when synchronization is completed was recorded. Although outliers were detected when two robots were not well aligned, these outliers were disregarded in this preliminary evaluation as it focuses on testing the algorithm itself by keeping a well-structured condition. With the developed algorithm with reduced packet length, the frequency of RSSI was increased. In very

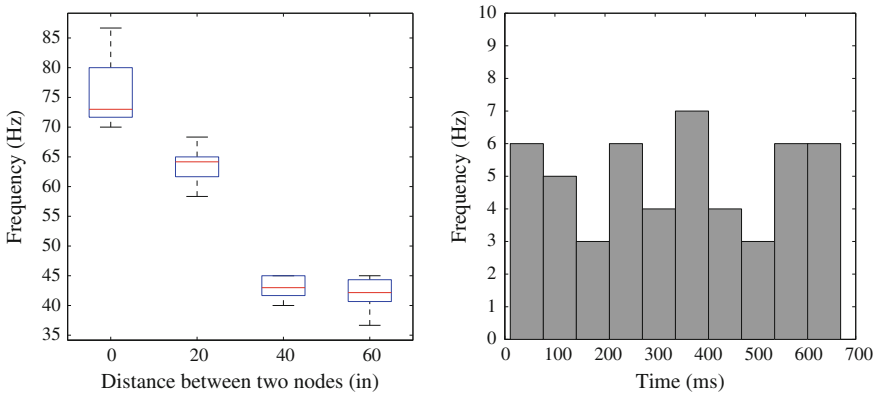


Fig. 5 Histogram of the self-synchronization time for 50 trials (Mean = 340 ms; SD = 200 ms). Frequency of measured RSSI by distance

close distance, frequency of updating RSSI recorded 70–80 Hz (Fig. 5). However, the frequency decreased by the distance. We initially expected the frequency would be consistent over distance as the radio signal travels with the speed of light, but in reality, the physical distance affected the frequency of RSSI collection.

3 Distributed Algorithms for Global Shape Formation

To demonstrate collective task execution by swarm robots with limited sensing and communication capabilities, we considered a group of swarm robots, each equipped with IR sensors for self-synchronization and a ZigBee module with about 60 in. of localization range. The swarm robotic system was modeled based on a virtual spring damper model. Algorithms for dispersion without a leader and line formation with an interim leader using only the distance estimation among the neighbors.

3.1 Dynamic Model of the Robotic Swarm

Our dynamic model of swarm systems involving multiple agents is based on a virtual spring damper model. To realize attractive and repulsive forces in a stable manner, a virtual damper is added to the spring system to suppress the oscillatory motion of the agents. Modeling dynamic multi-agent systems frequently requires global optimization by designing objective functions. As the number of agents increases, the computational cost for optimization increases exponentially. To address this problem, we used geometric conditions that guarantee global shape formation from geometric primitives based on the attractive and repulsive forces between the agents.

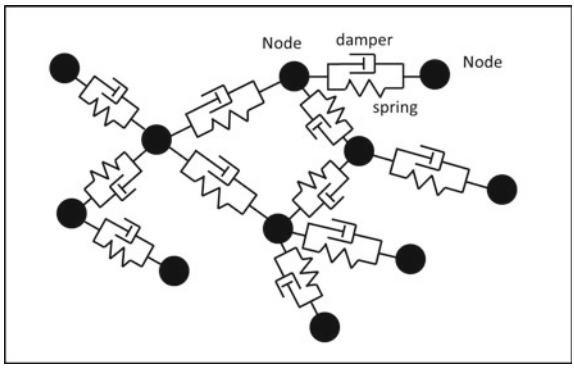


Fig. 6 Attractive and repulsive forces between the nodes are modeled with a spring and a damper

We consider a dynamic model of collective agents while maintaining connectivity using attractive and repulsive potential fields based on a mass, spring, and damper system as shown in Fig. 6. Each node and edge represent an agent and a communication/sensing/social connection respectively where the spring constant and damping ratio may be differently defined depending on each application. The mass of each node may indicate the weight or relative importance of that agent within the group.

Let $A_1, A_2, \ldots, A_n$ be the agents in two-dimensional space. For the ith agent, the two-dimensional Cartesian coordinate is denoted by $\boldsymbol{x}_i = [x_i, y_i]^T$. Likewise, the vector from the ith agent to the jth agent is given by $\boldsymbol{x}_{ij} = [x_i - x_j, y_i - y_j]^T$ where $||\boldsymbol{x}_{ij}|| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Assuming that the ith agent has a mass, m_i. Then it follows the Newton's second law: $\boldsymbol{f}_i^m = m_i \ddot{\boldsymbol{x}}_i$ where $\boldsymbol{f}_i^m$ is the force at the ith node and x_i is the position of ith node. The force exerted by the spring and damper is described by $\boldsymbol{f}_i^k = -k_{ij}\boldsymbol{d}_{ij}$ and $\boldsymbol{f}_i^b = -b_{ij}\dot{\boldsymbol{x}}_{ij}$ where k_{ij} is the spring constant, b_{ij} is the damping coefficient between agent i and neighboring agent j, and d_{ij} is a displacement vector between the two. In this modeling, orientation from the agent to another agent is uncertain since each agent measures the distances to other agents only. However, $\boldsymbol{d}_{ij}$ is expressed in a vector form since the sum of multiple interactions to an agent will force the agent to a certain direction in global coordinate system. Therefore, the net force acting on the ith agent in the global coordinate can be derived by above relationships:

$$\boldsymbol{f}_i^m = \boldsymbol{f}_i^k + \boldsymbol{f}_i^b = -\sum_{j \in M_i} (k_{ij}\boldsymbol{d}_{ij} + b_{ij}\dot{\boldsymbol{x}}_{ij})$$

where M_i is the number of neighboring agents of the ith agent. The acceleration of the ith agent is calculated as $\ddot{\boldsymbol{x}}_i = -\sum_{j \in M_i} (k_{ij}\boldsymbol{d}_{ij} + b_{ij}\dot{\boldsymbol{x}}_{ij})/m_i$. In practical control with a microprocessor, the above equation can be rewritten in a discrete time representation given by

$$\dot{\boldsymbol{x}}_i[t+1] - \dot{\boldsymbol{x}}_i[t] = \frac{-\sum_{j \in M_i} (k_{ij}\boldsymbol{d}_{ij} + b_{ij}\dot{\boldsymbol{x}}_{ij}[t])}{m_i}.$$

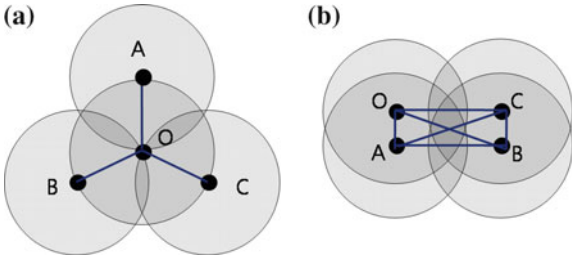


Fig. 7 Elementary geometric shapes: **a** trigonal planar and **b** paired line

In case of $m_i = 1$ for all $i = 1, \ldots, n$, the anticipated velocity at $t + 1$ and position at $t + 2$ are given by

$$\dot{\boldsymbol{x}}_i[t+1] = \dot{\boldsymbol{x}}[t] - \sum_{j \in M_i} (k_{ij}\boldsymbol{d}_{ij} + b_{ij}\dot{\boldsymbol{x}}_{ij}[t])$$

$$\boldsymbol{x}_i[t+2] = 2\boldsymbol{x}_i[t+1] - \boldsymbol{x}_i[t] - \sum_{j \in M_i} (k_{ij}\boldsymbol{d}_{ij} + b_{ij}\dot{\boldsymbol{x}}_{ij}[t]).$$

3.2 Dispersion and Paired Line-Formation Algorithms

Two algorithms are introduced here: (1) dispersion based on trigonal planar elements without a leader (Fig. 7a) and (2) line formation using paired line elements using an interim leader (Fig. 7b). We assume that only the distance data measured from the RSSI whine the sensing range is available.

Dispersion using trigonal planar elements: Let N be a set of dynamic agents. For every $O \in N$, label the closest three neighbor agents as A, B, and C. By using the proposed attractive and repulsive forces, let $L_{OA} = L_{OB} = L_{OC} = L_d$ where L_d is the desired distance between the agents.

Forming a line using paired line elements: Let N be a set of dynamic agents. For every $O \in N$, label the closest agent as A. Let $L_{OA} = \epsilon$ and form a pair, (O, A). Find the nearest pair and label as (B, C). Let $L_{OB} = L_{OC} = L_{AB} = L_{AC}$. Note that $L_{OB} = L_{OC}$ for $L_{OA} = L_{BC} = \epsilon$ as $\epsilon \to 0$. When all rectangles are connected to each other, two ends are designated as interim leaders of the connected chains in order to stretch the line to opposite directions.

3.3 Simulation Results

MATLAB simulations were performed with 50 agents randomly spread in two-dimensional space. The simple trigonal elements guarantee dispersion of the agents and paired line elements guarantees line shape while maintaining connectivity as

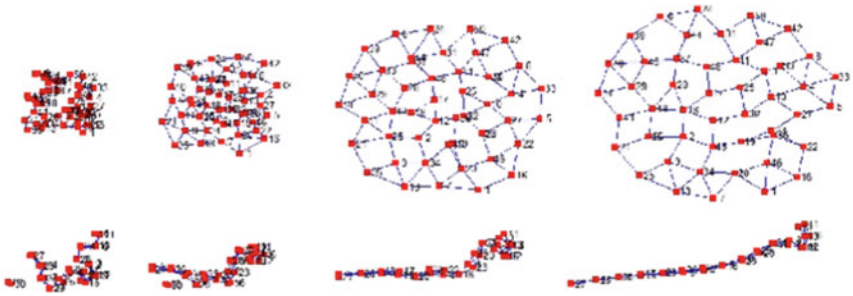


Fig. 8 Dispersion based on trigonal planar elements (*top*), Line formation using paired line elements (*bottom*)

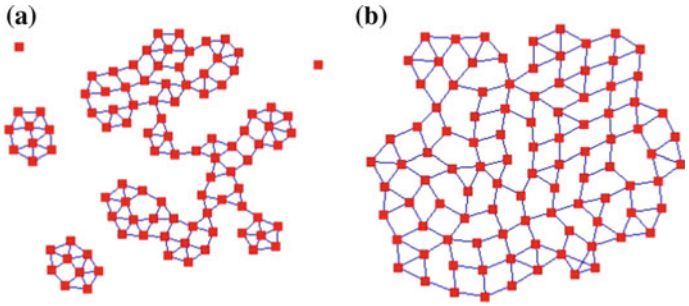


Fig. 9 Separated discs and connected disc with different sensing range: **a** 40 in., **b** 60 in. with 100 agents

shown in Fig. 8. If the communication range is less than 60 in., it is possible to form several smaller discs or lines resulting in outliers. Figure 9 shows two simulation results using two different communication ranges, 40 in. and 60 in.

4 Conclusion and Discussion

This paper presented wireless communication and distance sensing algorithms using the IR- and RF-based technologies. To realize decentralized communication in swarm robotics, we presented a self-synchronization technique using IR sensors between the adjacent robots. An algorithm using bi-directional RSSI achieved distance measurement between two robots. Also, we further presented two decentralized global shape formation algorithms such as dispersion and forming a line, and showed simulation results using 50 robots. As briefly discussed in the previous section, there exist several physical challenges in order to establish reliable IR/RF-based communication in the swarm robots (e.g. signal interference). Three main challenges are (1) unreliability of RSSI measurement over a long-distance, (2) interference of the RSSI signal when there are blocking objects or robots between the two, and (3) accurate

localization combining RSSI data and IR communication. These issues may be resolved by limiting the communication range to the range of reliable RSSI measurement range (up to 60 in.) despite a typical communication range of the XBee module is 300 ft. In addition, by combining the existing physical constraints into signal process, the faulty RSS data can be filtered out. For instance, if the robot moves up to a certain speed and the RSSI values measured between a certain time-step indicates a farther distance, this value may be discarded. Regarding the accurate localization with short IR coverage and noisy RSSI, dynamically variable IR coverage can be considered with digital potentiometer to increase the sensing range.

References

1. Bonabeau, E., Theraulaz, G.: Swarm Smarts, pp. 73–79. Scientific American, Inc. (2000)
2. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. 53–66 (2002)
3. Seeley, T.D., Camazine, S., Sneyd, J.: Collective decision-making in honey bees: how colonies choose among nectar sources. Behav. Ecol. Sociobiol. **28**, 277–290 (1991)
4. Navaro-Serment, L.E., Paredis, C.J.J., Khosla, P.K.: A beacon system for the localization of distributed robotic teams. In: Proceedings of International Conference on Field and Service, Robotics (1999)
5. Bisson, J., Michaud, F., Letourneau, D.: Relative positioning of mobile robots using ultrasounds. In: Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems, pp. 1783–1788 (2003)
6. Rivard, F., Bisson, J., Michaud, F., Letourneau, D.: Ultrasonic relative positioning for multi-robot systems. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 323–328 (2008)
7. Grabowski, R., Khosla, P.: Localization techniques for a team of small robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1067–1072 (2001)
8. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low-cost outdoor localization for very small devices. Pers. Commun. **7**(5), 28–34 (2000)
9. 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: Swarm Intelligence Symposium, pp. 37–44 (2007)
10. Mondada, F., et al.: The e-puck, a robot designed for education in engineering. IEEE ICARSC **1**(1), 59–65 (2009)
11. Lorincz, K., Welsh, M.: Motetrack: a robust, decentralized approach to RF-based location tracking. In: Location-and Context-Awareness, pp. 63–82 (2005)
12. Mondada, F., Franzi, E., Guignard, A.: The development of Khepera. In: Experiments with the Mini-Robot Khepera, Proceedings of the First International Khepera Workshop, No. LSRO-CONF-2006-060, pp. 7–14 (2006)
13. Dantu, K., Rahimi, M., Shah, H., Babel, S., Dhariwal, A., Sukhatme, G.S.: Robomote: enabling mobility in sensor networks. In: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, pp. 55 (2005)
14. Benkic, K., et al.: Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. In: IWSSIP, pp. 303–306 (2008)
15. Seybold, J.S.: Introduction to RF Propagation. Wiley Interscience, New York (2005)
16. Kim, M., Chong, N.Y.: Direction sensing RFID reader for mobile robot navigation. IEEE Trans. Autom. Sci. Eng. **6**, 44–54 (2009)

17. Jeong, D., Lee, K.: Directional RSS-based localization of multi-robot applications. In: 12th WSEAS, Cambridge, UK, Feb 2013
18. Pugh, J., Raemy, X., Falconi, R., Martinoli, A.: A fast on-board relative positioning module for multi-robot systems. IEEE Trans. Mechatron. **14**(2), 151–162 (2009)
19. Awad, A., Frunzke, T., Dressler, F.: Adaptive distance estimation and localization in WSN using RSSI measures. In: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, pp. 471–478 (2007)
20. Khaluf, Y., Micus, S., Weiss, F.: Master election for time synchronization in swarm robotic systems. In: 10th International Symposium on Parallel and Distributed Processing with Applications, pp. 285–292 (2012)
21. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. In: OSDI'02, vol. 36, pp. 147–163 (2002)
22. Lam, W.M., Reibman, A.R.: Self-synchronizing variable-length codes for image transmission. In: IEEE ICASSP, pp. 477–480 (1992)

Distributed Co-optimisation of Throughput for Mobile Sensor Networks

Trung Dung Ngo

Abstract We study the problems of throughput optimisation of mobile sensor networks. A network of mobile sensor nodes equipped with limited sensing and communication capabilities for connectivity maintenance and measurement of quality of communication links with the nearest neighbours is deployed to exploit and collect environmental data. Communication throughput of the multi-hop ad-hoc network of mobile sensor nodes is maximised for fast and reliable data transmission from sources to destinations. We propose a method of designing the distributed control for mobile sensor nodes for throughput optimisation in two stages: (1) position-aware optimisation and (2) communication-aware optimisation. We demonstrate effectiveness of the method through Monte-Carlo simulation based statistical results.

Keywords Mobile sensor networks · Throughput · Distributed co-optimisation · Communication-aware optimisation · Position-aware optimisation

1 Introduction

Mobile sensor networks have recently received significant attentions due to their potential applications. This new research field is roughly considered as the intersection of the two well established fields, wireless sensor networks and multi-robot systems, but enriched with diversification of integrating sensing awareness, communication awareness into mobility control for mobile sensor nodes in order to enhance system performances.

A mobile wireless sensor network of mobile sensors obtains several advantages over a traditional wireless sensor network due to controllable mobility of mobile sensors. For instances, a small number of mobile sensors can cooperatively navigate to explore and cover large hazardous environmental areas, detect and exploit

T.D. Ngo (✉)
The More Than One Robotics Laboratory, Faculty of Science,
University of Brunei Darussalam, Bandar Seri Begawan, Brunei Darussalam
e-mail: dungnt@ieee.org
URL: http://www.morelab.org

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_29

environmental data with various equipped sensors, and transmit exploited data to human operators using a multi-hop network of mobile relays. However, we encounter more challenges when deploying and managing a mobile sensor network rather than a transitional wireless sensor network due to spatio-temporal dynamics of mobile sensor nodes, i.e., connectivity maintenance, network topology, data transmission. On the other hand, typical aspects of communication networks, such as communication channels, routing protocols, quality of services have not often considered as research problems in multi-robot systems, instead it is assumed as being available for facilitating the development of various applications, i.e., coordinative exploration, cooperative coverage, collaborative environmental patrolling and monitoring. Connectivity maintenance/preservation—the concept often mentioned in multi-robot systems—is mostly considered in terms of sensing connectivities rather communication connectivities (communication links), and its roles in guaranteeing quality of data communication in a multi-hop network of mobile nodes used to facilitate many applications of networked robotic systems has been not significantly studied. Nevertheless, the quality of communication links is the key element to guarantee inter-communication among sensor agents for their cooperative, coordinative, and collaborative operations. In this paper, we address comprehensive understanding of how to *integrate advantageous properties of wireless sensor networks, i.e., communication awareness, and multi-robot systems, i.e., localisation awareness, to enhance the system performance in terms of communication throughput*.

1.1 Literature Review

Graph theoretic network: graph theory is widely utilised as an useful tool to model networked systems including wireless sensor network and networked robotic systems. Agents (either robotic or sensor agents) and their connectivities in a networked system are mathematically modelled by nodes and edges of either directed or undirected graph. The Laplacian matrix-based algebraic connectivity is often used to check the global connectivity of the network. Coordinative and cooperative operations of networked systems can be modelled, controlled, and optimised using graph theories and properties [1–5]. However, the primary drawback of representing a multi-agent network in graph theory is that the second smallest eigenvalue—the Fielder value—of the Laplacian matrix is not differentiable so that it is not possible to design feedback control for connectivity maintenance. Connectivity of pairs of agents is therefore equipped with either linear [6] or non-linear weights [2] that works as potential functions to facilitate the feedback control design and stability validation [7–10].

Artificial potential field: the artificial potential field, coined out by Khatib [11], is the well-known method developing artificial potential force-based control by synthesising attractive and repulsive forces. This method is purely based on the local

sensing and perception of mobile agents about their peers and environments to drive the mobile robots towards the goal without colliding with obstacles. The artificial potential field method has been extended to develop decentralised mobility control for mobile agents in multi-agent systems (including multi-robot systems and mobile sensor networks) [3, 12–17].

Mobility in wireless networks: Research on impacts of mobility in wireless networks has been traditionally investigated along the other known wireless communication issues causing reduction of quality of communication services i.e., multi-path falling, shadowing, interferences, and Doppler phenomenon. In contrast, mobility has been increasingly considered as an impact factor enhancing usabilities of wireless networks [18–23].

1.2 Motivation

In the real-world applications, a mobile wireless sensor network is deployed into an unknown environment for exploration and environmental data exploitation through two stages: *network deployment*, and *network utilisation*. In the first stage, the wireless network of mobile sensors is sent into the environment to self-organise a multi-hop network used to transfer exploited data to human operators. In the second stage, once the multi-hop wireless network is established through interconnected mobile sensor nodes, communication throughput of the network should be maximised for faster and more reliable data transmission from sources to destinations. Controllable mobility of mobile nodes can be utilised to improve the communication throughput of the wireless ad-hoc network. However, this research direction has not been significantly studied, to the best of our knowledge.

In the scope of this paper, we assume that the first stage of the *network deployment* has been done since mobile sensor nodes have been deployed in the environment. We primarily consider the second stage of the *network utilisation* with an emphasis on throughput optimisation of mobile sensor networks by incorporating two distributed optimisation techniques: ***position-aware optimisation*** and ***communication-aware optimisation***. Specifically, the distributed co-optimisation is developed by integrating advantages of artificial potential fields maintaining sensing and communication connectivities of mobile sensor nodes, and the Max-Flow-Min-Cut graph theory representing communication capacity and information flows of communication links.

1.3 Contributions of This Paper

This paper provides a novel method of distributed co-optimisation of throughput for mobile sensor networks. The contributions can be seen in twofolds:

- Enforcing two research fields, wireless sensor networks and multi-robot systems, closer by integrating advantages of artificial potential force field and the Max-Flow Min-Cut graph theorem to design the distributed control of mobile sensors for throughput optimisation.
- Realising a method of distributed co-optimisation—an event-triggered optimisation—based on information flows between mobile sensor notes. This method is fully distributed, allowing a mobile sensor network to enhance its convergence, adaptability, and scalability.

1.4 Paper Outline

The rest of this paper is organised as follows. A short tutorial on graph-theoretic network model of the Max-Flow Min-Cut theorem is described in Sect. 2. Preliminaries of research statement and motivation are in Sect. 3. Detailed description of distributed co-optimisation method consisting of position-aware optimisation and communication-aware optimisation is explained in Sect. 4. Monte-Carlo simulation based statistical results are shown and discussed in Sect. 5. We conclude this paper with future research directions in Sect. 6.

2 Graph-Based Network Model—A Short Tutorial

A graph *G(V, E)* is employed to describe a multi-hop wireless ad-hoc network of mobile sensor nodes. V is defined as collection of mobile sensor nodes while E is denoted for the set of peer-to-peer communication links between them. V is divided in three groups: sources S—sending data packets, sinks T—receiving data packets, and routers R—relaying data packets from a source to a destination, where S and T must be non-empty. Each edge $e(v_i, v_j) \in E$ represents a communication link with a nonnegative capacity $c(v_i, v_j) \geq 0$. Actual data flow $f(v_i, v_j)$ between two mobile nodes v_i and v_j must be less than capacity of the communication link, $f(v_i, v_j) \leq c(v_i, v_j)$ for every $v \in V \backslash \{s, t\}$. On a node $r \in R$, inflow data must be equal to outflow data, $f_i(r) = f_o(r)$, where inflow $f_i(r)$ is the sum of incoming data into the node while outflow $f_o(r)$ is the total data coming out from such a node. Data packets can be transferred from a source $s \in S$ to a destination $t \in T$ through either single or multiple communication channels in the wireless ad-hoc network of mobile nodes. The value of a flow f, denoted $val(f)$, from a source s is the amount of data sending out from a source s to a sink t: $val(f) = \sum_{v \in V} f(s, t)$. *Maximum flow*, denoted f^{max}, is the maximum network flow: $val(f^{max}) \geq val(f), \forall f$. A *cut* is a group of edges connecting a group of sources V_s and a group of destinations V_t through a number of relay nodes $V_r \in V$. Capacity of a *cut* is the total capacity of communication links on the *cut*, $c(V_s, V_t)$. *Minimum cut*, denoted $c^{min}(V_s, V_t)$, of the network is the minimum total capacity of the set of communication links involved in the *cut*.

Bottlenecks of a network are where communication links with *maximum flows* are existing. According to the Max-Flow Min-Cut graph theorem [24], *maximum flows* are *minimum cuts* so that we can search for *minimum cuts*, instead *maximum flows*, to find bottlenecks of a network. The *residual graph*, $G(V, E) : c_f(e_j, e_i) = c(e_i, e_j) - f(e_i, e_j)$ is used to check whether an *augmenting path* exists in the residual graph *G(V, E)* connecting a source S to a destination T. No existence of augmenting path in the *residual graph* guarantees that the network is at maximum flow. Note that, if the network has more than one source (destination), a super-source (super-destination) connecting with all sources (all destinations) by edges of *infinity* capacity is required for generating the *residual graph*.

3 Preliminaries and Problem Statement

3.1 Relative Localisation and Communication Capacity

We consider a kind of wireless networks of mobile sensor nodes communicating each other in the peer-to-peer fashion. The network is deployed into an unknown environment for fast exploration, monitoring, patrolling, and data collection. The mobile sensor nodes automatically navigate to explore in the environment while preserving connectivities with their neighbouring peers. Mobile sensor nodes are capable of self-organising a multi-hop wireless network for information exchange. A node becomes a source if it transfers environment-exploited data through the self-organising network of mobile routers—working as relays of the network—to a destination. Data packets are delivered through either single or multiple communication channels of the wireless network before reaching the destination. Therefore, communication throughput of such channels is expectedly maximised for faster and more reliable data transmission.

We assume that the mobile sensor nodes are equipped with ranging sensors that can sense and measure the relative distance to other sensor nodes and obstacles within its local vicinity for their manoeuvrability. Without loss of generality, we presume that the communication range is longer than the sensing range for all mobile sensors so that two sensor nodes can communicate well if they are mutually within their sensing range. In other words, communication connectivity can be identified through sensing connectivity. According to the principle of path-loss of signal propagation (without considering effectiveness of multi-path fading and shadowing as explained in [25, 26]), capacity of a communication link established by two mobile sensor nodes is inversely proportional to their relative distance so that capacity of a communication link of two mobile sensor nodes increases if such mobile sensor nodes manoeuvre towards each other. However, this principle can not apply for the actual information flow because it depends on usage of such a communication link in the network—a communication link is established between two nodes but no actual information is delivered throughout this link due to routing protocols.

Relative localisation of the mobile sensor nodes can be identified through estimation of relative distance and relative bearing extracted by the ranging sensors. Artificial potential force field (APF) coined out in [11] is the popular method maintaining relative localisation of mobile sensor nodes by the trade-off of attractive forces pulling the nodes closer and repulse forces pushing the nodes away. Artificial Physics-based potential force field introduced in [14] is a simple but efficient method for preserving sensing connectivities of mobile sensor nodes using a combination of relative distance and relative bearing. However, the Artificial Physics-based mobility control might not directly optimise communication throughput of the network because this mobility control is not aware of capacity and information flow of communication links. Note that the Artificial Physics-based mobility control might implicitly improve capacity of communication links since the potential forces drive mobile sensor nodes to equilibrium positioning among their neighbouring nodes.

Communication-aware mobility control on a sensor node is expected to relocate the node to appropriate positioning among their nearest neighbours where they contribute better to the network throughput. To do so, this sensor node must be aware of its relative localisation and capacity and information flow of communication links made with its nearest neighbours. However, on one hand, measures of capacity and actual information flow of communication links are not sufficient enough to design the mobility control of mobile sensor nodes due to lacking information of relative localisation because most of communication mechanisms used for wireless sensor networks is a kind of unidirectional signal propagation. On the other hand, sensing-based relative localisation is not sufficient enough to design mobility control of mobile sensor nodes for throughput optimisation as this control is not aware of capacity and information flow of communication links. As a result, we synthesise sensing-based relative localisation and measures of capacity and information flows of communication links to design the distributed mobility control of mobile sensor nodes that is capable of improving the network throughput.

3.2 Shortcomings of the Max-Flow Min-Cut Theorem in Distributed Schemes

The Max-Flow Min-Cut theorem cannot be directly applied to search for bottlenecks of a mobile sensor network due to the following shortcomings: (1) information of the entire network about the nodes and communication links must be deterministic; (2) computational complexity is high according to the centralisation scheme; (3) a super-source(-destination) is required if more than one communication channel exist. Indeed, in the Max-Flow-Min-Cut theorem [24], *maximum flows* are identified through capacity of *minimum cuts*, $f^{max}(V_s, V_t) = c^{min}(V_s, V_t)$, in a network, so that we can search for *minimum cuts* instead *maximum flows* in a multi-hop wireless network of mobile routers. However, to search for *minimum cuts* of a graph representing a network of mobile sensor nodes using the Max-Flow Min-Cut theorem, the

database of all the nodes and their communicational links must be deterministic for any heuristic searching algorithms applied for finding an *augmenting path*. That is, all the mobile nodes must exchange information about communication capacity and information flows of their communication links through the entire network, which is usually not applicable to wireless networks of mobile sensor nodes due to dynamic changes in terms of relative localisation and communication link capacity. Moreover, if multiple sources (destinations) exist, a virtual super source (destination) must be created and all sources (sinks) are connected to this super source (destination) with ∞ capacity of communication links to make a multi-commodity source (destination) for augmenting path algorithms, which is not often implementable for wireless networks in the real world. Computational complexity of the Max-Flow Min-Cut theorem is dependent to number of nodes and communication links in the network, i.e., Ford-Fulkerson algorithm $O(E|f|)$, Edmonds-Karp algorithm $O(VE^2)$, Dinitz blocking flow algorithm $O(V^2E)$, as so the more nodes and communication, the higher computational cost. As a result, a large-scale network of mobile sensor nodes is not scalable to number of nodes as the sensor nodes encounter highly computational cost while a part of network bandwidth is reserved for transferring updating information of communication capacity of communication links and actual information flows from all other sensor nodes to every node for its operations of searching for *minimum cuts* of the network.

4 Distributed Co-optimisation

The great advantage of the Max-Flow Min-Cut theorem is to find bottlenecks—*minimum cuts*—of a graph-modelled network. Without this algorithm, we cannot find *minimum cuts* where the mobile sensor nodes should move towards in order for increasing capacity of the existing communication links. However, to employ the Max-Flow Min-Cut algorithm for throughput optimisation of mobile sensor networks, we encounter the limitations which are not feasible in distributed schemes as explained in Sect. 3.2.

We propose a distributed co-optimisation that decentralises the Max-Flow Min-Cut theorem by searching for *possible minimum cuts*, where there highly possibly exist saturated communication links, $f(v_i, v_j) \leq c(v_i, v_j)$. The mobile sensor nodes are capable of estimating *possible minimum cuts* by measuring differential between capacity and information flow of communication links made with their neighbouring nodes, $c(v_i, v_j) - f(v_i, v_j)$. Such nodes also need to identify relative directions of *possible minimum cuts* and maintain relative localisation (relative distance and relative bearing) with their neighbouring nodes through the Artificial Physics-based potential forces. As a result, we synthesise capacity and information flows of communication links and the Artificial Physics-governed relative localisation to develop

a distributed mobility control of mobile sensor nodes for throughput optimisation as seen in (1):

$$F_{dco}(v_i) = \begin{cases} \sum\limits_{v_j \in N(v_i)} \dfrac{f(v_i, v_j)}{\varepsilon + (c(v_i, v_j) - f(v_i, v_j))} * \dfrac{\overrightarrow{F_{AP}}(v_i, v_j)}{\| \overrightarrow{F_{AP}}(v_i, v_j) \|} & \text{if } f(v_i, v_j) \neq 0 \\ \sum\limits_{v_j \in N(v_i)} \overrightarrow{F_{AP}}(v_i, v_j) & \text{if } f(v_i, v_j) = 0 \end{cases} \tag{1}$$

where $c(v_i, v_j)$ and $f(v_i, v_j)$ is the capacity and information flow of communication link $e(v_i, v_j)$ made by the sensor node v_i with its neighbouring nodes, $v_j \in N(v_i)$, respectively; $\overrightarrow{F_{AP}}(v_i, v_j) = \frac{G * m(v_i) * m(v_j)}{r^2}$ is the Artificial-Physics-based potential force between the sensor node v_i and its neighbouring nodes, $v_j \in N(v_i)$, in which r is the relative distance between i and j and the gravitational constant G is arbitrarily chosen; ε is arbitrarily chosen as small as possible, $\varepsilon \ll \sum_{v_j \in N(v_i)} c(v_i, v_j)$, to avoid the case of fully saturated communication links, $c(v_i, v_j) = f(v_i, v_j), \forall v_j \in N(v_i)$.

All the mobile sensor nodes with the distribtued mobility control described in (1) operate in two stages:

- **Position-Aware Optimisation**: If no information flow is detected by the mobile sensor node, it only uses the Artificial Physics-based potential forces to maintain connectivities with their neighbouring nodes, which might indirectly impact on improvement of capacity of communication links leading to better throughput.

$$F_{pao}(v_i) = \sum_{v_j \in N(v_i)} \overrightarrow{F_{AP}}(v_i, v_j) \quad \text{if } f(v_i, v_j) = 0 \tag{2}$$

- **Communication-Aware Optimisation**: If information flows are detected by the mobile sensor node, it uses *possible minimum cut* based potential force F_{cao} to navigate towards the neighbouring node involved in the *minimum cut* with the most saturated communication links in order for gaining capacity of communication links leading to better throughput (Fig. 1).

$$F_{cao}(v_i) = \sum_{v_j \in N(v_i)} \frac{f(v_i, v_j)}{\varepsilon + (c(v_i, v_j) - f(v_i, v_j))} * \frac{\overrightarrow{F_{AP}}(v_i, v_j)}{\| \overrightarrow{F_{AP}}(v_i, v_j) \|} \quad \text{if } f(v_i, v_j) \neq 0 \tag{3}$$

5 Monte-Carlo Simulations and Discussions

5.1 Experiment Setup and Performance Metrics:

The Monte-Carlo simulation method is applied to generate randomised experimental scenarios. A typical scenario with three stationary base stations operated as sources

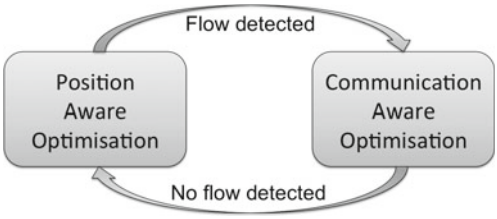


Fig. 1 Distributed co-optimisation for mobile sensor networks—information flow triggered optimisation—in two stages: if the information flows are detected, the communication-aware optimisation is activated to relocate mobile sensor nodes to new positions with better throughput through Eq. 3; if information flows are not detected, the position-aware optimisation is used to preserve the communication connectivities among the sensor nodes, which might implicitly improve throughput through Eq. 2

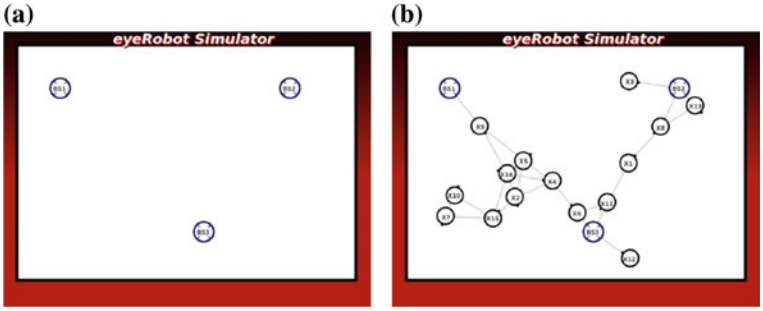


Fig. 2 The experimental scenario with three stationary base stations (*LHS*) stationary base stations, (*RHS*) randomly placed mobile robots. **a** Three stationary *base stations*. **b** 15 randomly placed *sensor nodes*

and destinations are created as seen in Fig. 2a. We have applied the Gaussian random distribution to place the sensor nodes into the experimental scenario as an example illustrated in Fig. 2b. The generated scenarios are selected for experiments if three base stations are well connected through an ad-hoc wireless network of mobile sensor nodes. For each experiment, we executed 10000 simulation steps to measure the network throughput between the base stations. The statistical results shown in Fig. 3 were collected from 100 randomised scenarios.

We propose four key performance metrics to evaluate the developed distributed co-optimisation algorithm:

- **Optimality**: how much is the network throughout of the mobile wireless sensor network gained over time?
- **Adaptability**: is the network throughput adaptable to incremental number of sensor nodes added into the network?
- **Convergence**: how fast does the network throughput converge to a steady state?
- **Scalability**: does computational complexity of sensor nodes increase to infinity if the number of sensor nodes added into the network increases to infinity?

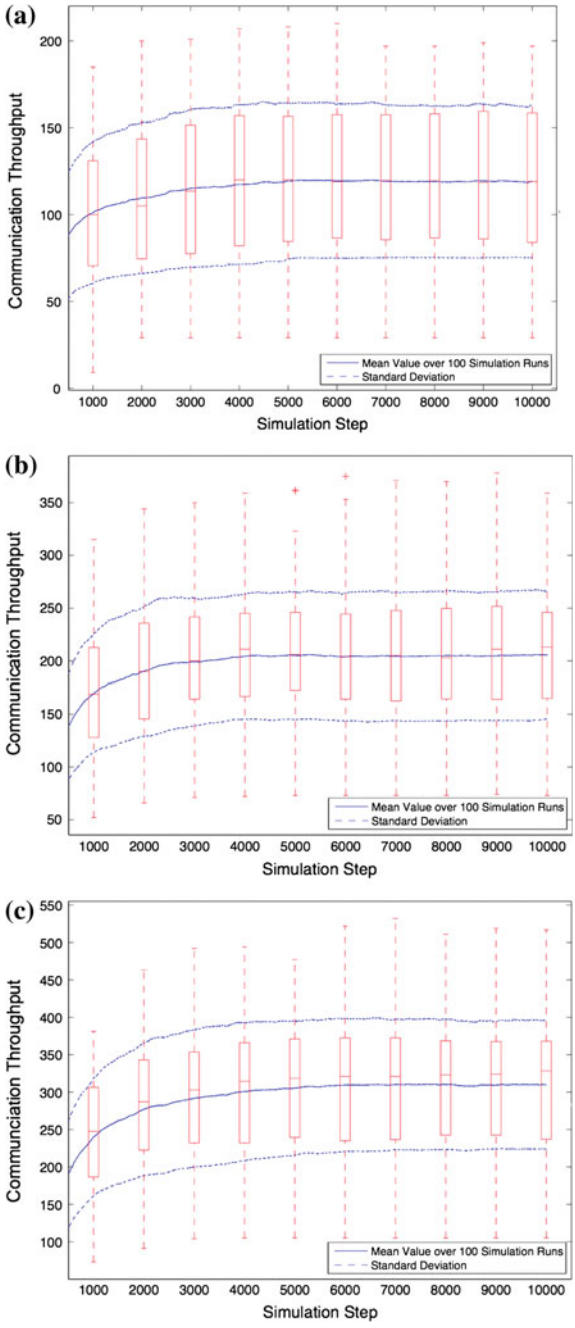


Fig. 3 The Monte-Carlo simulation based statistical results of 3 stationary base stations. **a** 10 sensor nodes **b** 15 sensor nodes **c** 20 sensor nodes

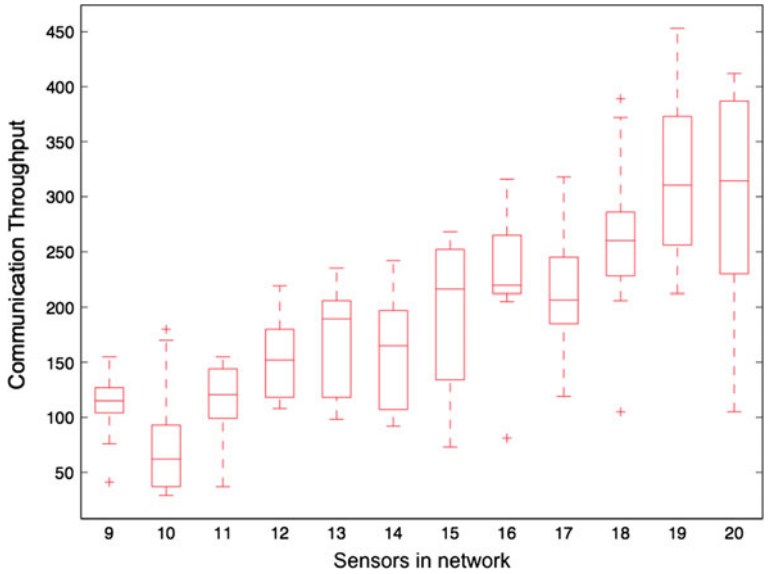


Fig. 4 Statistical results of random scenarios with the number of sensor nodes increased from 9 to 20

5.2 Results and Discussions

Based on the statistical results, we discuss on the key performance metrics of the distributed co-optimisation algorithm.

Optimality: Looking at the statistical results shown in Figs. 3 and 4, the network throughput is improved at all cases, which affirms that the distributed co-optimisation algorithm is capable of locally optimising the overall network throughout of mobile sensor networks.

Adaptability: The statistical results shown in Figs. 3 and 4 show that the network throughput of mobile sensor networks is gained with the improvement rate from 15 to 50 % approximately when the number of sensor nodes used in the network increases from 10 to 20 nodes. It is proved that the distributed co-optimisation adaptively deals with the number of sensor nodes used in the network.

Convergence: Looking into the statistical results of the network throughput in Fig. 3, the distributed co-optimisation algorithm enables the network throughput of mobile sensor networks converge to a steady state after 5000 running steps approximately in all cases. Convergence is one of the most important issues of mobile sensor networks because the sensor nodes no longer need to consume energy for their mobility control, and the communication channels become stable for data transmission between the base stations.

Scalability: Supposing that we use a centralisation method, i.e., Ford-Fulkerson algorithm, Edmonds-Karp, or Dinitz blocking flow algorithm, to search for *minimum-*

cuts, a sensor node must collect all information about capacity of communication links and information flows from the other sensor nodes through the network communication. This leads to dramatically increased computational cost on every sensor nodes i.e., Ford-Fulkerson algorithm $O(E|f|)$, Edmonds-Karp algorithm $O(VE^2)$, Dinitz blocking flow algorithm $O(V^2E)$ as well as an substantial amount of the network bandwidth reserved for updating such information (depending the number of sensor nodes deployed in the network). Applying the distributed co-optimisation algorithm, each mobile sensor node only uses its local information of information flows and capacity of communication links made with its nearest neighbours, as so no network bandwidth is reserved for updating such information of the network. Moreover, the computational complexity of mobile sensor nodes is almost constant, $O(1)$, as it is only dependant to the number of its nearest neighbours which is usually a small number in the real world.

Overall, the distributed co-optimisation method is reliably practical in the real-world applications because it enables mobile sensor networks to be adaptable in dynamic environments where communication links of mobile sensor nodes do not often last long due to mobility of mobile sensor nodes and uncertainty of environmental conditions, and scalable with the number of sensor nodes according to applicable requirements.

6 Conclusions and Future Directions

We have addressed the problems of throughput optimisation in mobile sensor networks. The distributed co-optimisation algorithm used to control mobility of mobile sensor nodes for throughput optimisation is developed by cross-fertilising advantages of the artificial potential force field and the Max-Flow Min-Cut graph theorem. The concept *possible minimum-cuts* is proposed as the key factor to optimise the throughput using the distributed optimisation algorithm. Through the Monte-Carlo simulations, we have proved that the developed algorithm have achieved all the four performance metrics: optimality, adaptability, scalability, and convergence.

In the near future, we are going to work on realistic communication models of communication links with considerations of multi-path fading, shadowing, and interferences as discussed in [26, 27] and even delay models of mobile relays before we call back this distributed co-optimisation method to validate system performances. We believe that network throughput might vary according to new parameters introducted, but this distributed optimisation method is highly applicable due to its systematic characteristics in terms of adaptability, scalability and convergence.

Acknowledgments This research was supported in part by the University Research Grant (UBD/PNC2/2/RG/1(259)).

References

1. Tanner, H., Jadbabaie, A., Pappas, G.J.: Stable flocking of mobile agents, part I: fixed topology. In: IEEE Conference on Decision and Control, pp. 2010–2015 (2003)
2. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Stable flocking of mobile agents, part II: dynamic topology. In: IEEE Conference On Decision And Control, pp. 2016–2021 (2003)
3. Kim, D.H., Wang, H., Shin, S.: Decentralized control of autonomous swarm systems using artificial potential functions: analytical design guidelines. J. Intell. Robotic Syst. **45**(4), 369–394 (2006)
4. Ji, M., Egerstedt, M.: Distributed coordination control of multiagent systems while preserving connectedness. IEEE Trans. Robot. **23**(4), 693–703 (2007)
5. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Automat. Control **49**(9), 1520–1533 (2004)
6. Olfati-saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Trans. Autom. Control **51**, 401–420 (2006)
7. Dimarogonas, D.V., Kyriakopoulos, K.J.: Connectedness preserving distributed swarm aggregation for multiple kinematic robots. IEEE Trans. Robot. **24**(5), 1213–1223 (2008)
8. Stump, E., Jadbabaie, A., Kumar, V.: Connectivity management in mobile robot teams. In: ICRA, pp. 1525–1530 (2008)
9. Schwager, M., Rus, D., Slotine, J.J.: Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. Int. J. Robot. Res. **30**(3), 371–383 (2011)
10. Tu, Z., Wang, Q., Qi, H., Shen, Y.: Flocking based distributed self-deployment algorithms in mobile sensor networks. J. Parallel Distrib. Comput. **72**(3), 437–449 (2012)
11. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986)
12. Elkaim, G.H., Kelbley, R.J.: Extension of a lightweight formation control methodology to groups of autonomous vehicles. In: ISAIRAS. Muchen (2005)
13. Reif, J.H., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. Robot. Auton. Syst. (1999)
14. Spears, D.F., Hamann, J.C., Heil, R.: Distributed, physics-based control of swarms of vehicles. Auton. Robots **17**, 137–162 (2004)
15. Ge, S.S., Fua, C.H.: Queues and artificial potential trenches for multi-robot formations. IEEE Trans. Robot. **21**(4), 646–656 (2005)
16. Andrew Howard, M.J.M., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, pp. 299–308 (2002)
17. Mikkelsen, S.B., Jespersen, R., Ngo, T.D.: Probabilistic communication based potential force for robot formations: a practical approach. In: DARS, pp. 243–253 (2010)
18. Grossglauser, M., Tse, D.N.C.: Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Trans. Netw. **10**(4), 477–486 (2002)
19. de Moraes, R.M., Sadjadpour, H.R., Garcia-Luna-Aceves, J.J.: Mobility-capacity-delay trade-off in wireless ad hoc networks. Ad Hoc Netw. **4**(5), 607–620 (2006)
20. Mostofi, Y.: Decentralized communication-aware motion planning in mobile networks: an information-gain approach. J. Intell. Robot. Syst. **56**(1–2), 233–256 (2009)
21. Ngo, T.D.: Linkmind: link optimization in swarming mobile sensor networks. Sensors **11**(8), 8180–8202 (2011)
22. Seol, J.Y., Kim, S.L.: Node mobility and capacity in wireless controllable ad hoc networks. Comput. Commun. **35**(11), 1345–1354 (2012)
23. Natalizio, E., Loscrì, V.: Controlled mobility in mobile sensor networks: advantages, issues and challenges. Telecommun. Syst. **52**(4), 2411–2418 (2013)
24. Ford, L., Fulkerson, D.: Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
25. Haenggi, M.: Analysis and design of diversity schemes for ad hoc wireless networks. IEEE J. Sel. Areas Commun. **23**(1), 19–27 (2005)

26. Liu, X., Haenggi, M.: Throughput analysis of fading sensor networks with regular and random topologies. EURASIP J. Wirel. Commun. Netw. **2005**(4), 554–564 (2005)
27. Fida, A., Iqbal, M., Ngo, T.D.: Communication—and position-aware reconfigurable route optimization in large-scale mobile sensor networks. EURASIP J. Wirel. Commun. Netw. **2014**, 207 (2014)

Detection and Notification of Failures in Distributed Component-Based Robot Applications Using Blackboard Architecture

Michael Shin, Taeghyun Kang and Sunghoon Kim

Abstract This paper describes detection and notification of component failures in distributed component-based robot applications using the blackboard architecture. The blackboard architecture monitors each component of robot applications in order to detect component failures at runtime and it identifies the causes of failures. Using the dependency relationships between components, the blackboard architecture performs impact analysis between components so that it determines the scope of failure notification in the components of a distributed robot application. The notification messages delivered to components can trigger actions against the failures if robot application developers have implemented the actions along with application functions. The prototype of blackboard architecture has been implemented for the Microsoft Robotics Developer Studio (MSRDS) environment, and it has been applied to the Unmanned Ground Vehicle (UGV) application implemented on the simulator as a case study.

Keywords Detection · Notification · Blackboard architecture · Robot component · Impact analysis

M. Shin (✉)
Department of Computer Science, Texas Tech University, Lubbock, TX, USA
e-mail: michael.shin@ttu.edu

T. Kang
Department of Computer Science, Wake Forest University, Winston-salem, NC, USA
e-mail: kangth@wfu.edu

S. Kim
Intelligent Robot Control Research, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea
e-mail: saint@etri.re.kr

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_30

1 Introduction

As robots are getting used in various application areas, more and more distributed robot applications are increasingly built with robot components. A robot component is a software unit that is functionally modularized and self-contained, that is, it can be compiled, instantiated, and linked separately into a distributed robot application [1]. Mission-critical, safety-critical, or business-critical robot applications, such as an unmanned ground vehicle (UGV) application, are being built with robot components for several advantages—abstraction, reusability, and maintainability. Also, component-based robot application development can reduce the cost and delivery time to market [2].

In spite of the advantages of robot components, distributed component-based robot applications [3, 4] need to be resilient against runtime failures, such as unanticipated sensor failures or periodic data failures. Unlike traditional industrial robots, distributed robot applications for intelligent robots should run under uncertain and dynamic environment and, in many cases, without human intervention. It is well known that runtime failures arising from robot components may not be discovered until all the components of an application are deployed together.

Several frameworks and platforms [5–13] have been proposed to develop robot applications using robot components in a systematic manner. Some of these approaches may provide the mechanisms for handling low level faults. However, the approaches to supporting component-based robot applications do not address detection of runtime component failures and notification of those to the associated components of robot applications. It is necessary for an approach to managing runtime component failures so that the distributed component-based robot applications are resilient to failures.

This paper describes an approach for detection and notification of robot component failures to the associated components of robot applications using the blackboard architecture. The blackboard architecture monitors each component of robot applications in order to detect component failures at runtime. Once the blackboard architecture detects a failure, it notifies the failure to the associated components. The proposed approach enables robot application developers to implement the actions against each failure, which can make the application resilient to the failures. The prototype of blackboard architecture has been developed for the Microsoft Robotics Developer Studio (MSRDS) environment, and it has been applied to the Unmanned Ground Vehicle (UGV) application implemented on the MSRDS simulator.

2 Blackboard Architecture

Figure 1 depicts the blackboard architecture that manages failures in component-based robot applications. The blackboard architecture is structured to failure/recovery monitor, failure/recovery notifier, and failure/recovery repository.

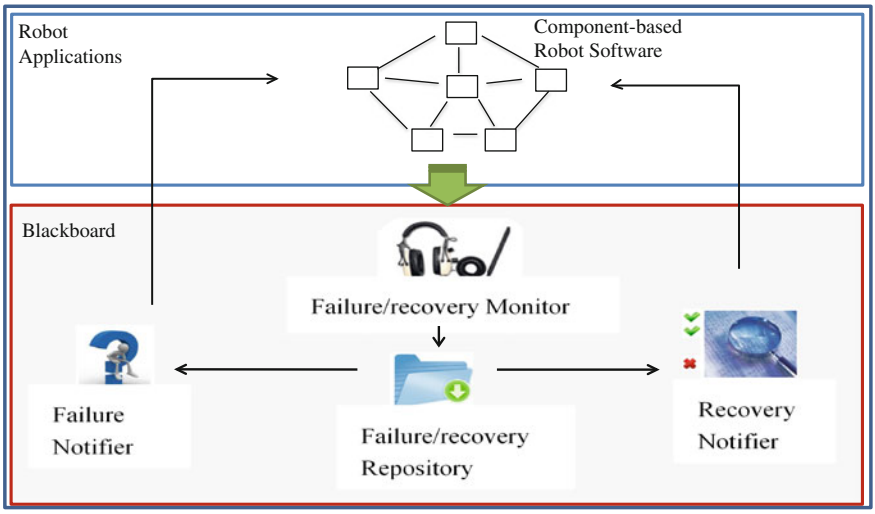


Fig. 1 Blackboard architecture

- Failure/Recovery Monitor. The blackboard architecture monitors components of robot applications and detects the runtime failures of components. The typical features of robot applications are sensor and actuator interaction, periodic data handling, real-time response to sensor inputs, and communication between distributed robot components. The blackboard architecture is capable of monitoring and detecting the failures of these features at runtime, analyzing them to find out which component is the cause of the failures. Also, the blackboard architecture can recognize the recovery of failed components.
- Failure/Recovery Notifier. The blackboard architecture provides notification messages to the components associated with a component failure. For this, the blackboard architecture maintains the dependency relationships between components of each application, performing failure impact analysis in order to determine the scope of notification. The notification messages delivered to components may trigger the failure handling actions automatically if robot application developers have implemented the actions in the components. In addition, if a failure is recovered, the blackboard architecture updates the components associated with the recovery so that the components resume their failed services.
- Failure/Recovery Repository. The blackboard architecture contains the failure/recovery repository to maintain the status of components. The failure/recovery repository includes the name of failed/recovered components and their failure types.

Although we assume that a robot application is designed and implemented by means of robot components, the blackboard architecture does not make any

assumption about component models for applications. A robot component can be one that is developed with or without a formal robot component model like MSRDS or Robot Operating System (ROS).

3 Failure/Recovery Monitor

3.1 Failure/Recovery Functions

The blackboard architecture provides component-based robot applications with detection functions, which can detect runtime failures of components. The types of runtime failures that can be detected by the blackboard architecture are as follows:

- Sensor failure: A sensor failure occurs when a sensor cannot measure values in the environment periodically. The sensor failure is detected by a sensor interface component by means of either the sensor input arrival rate or sensor input sampling rate. The blackboard architecture detects a sensor failure if the sensor input values are not delivered to a sensor interface component periodically.
- Sensor measurement range failure. The values that a sensor reads in the environment may be out of the range specified for the sensor. It may not be easy to detect sensor measurement range failures because there are many different brands for each sensor type. However, this failure can be detected if the sensor input value range is known.
- Real-time operation failure. When an operation provided by a robot component is invoked, it may not finish its execution within a specified time interval. The execution time of an operation is measured to check if the operation meets at least the worst case of its hard real-time deadline. The execution time for an operation is calculated with the operation's starting time and finishing time.
- Operation parameter range failure. An operation in a robot component may be invoked with the wrong values of parameters. Before an operation executes, the values of parameters for the operation are checked if they are in the ranges specified.
- Operation output range failure. An operation in a robot component may produce an output that is out of the specified range. The operation output range failure is detected by comparing an operation output to the range specified.
- Periodic data failure. Most of the robot components receives data from either sensors or other components periodically and then processes them. A data receiver component cannot process the data if a sender component does not send the data to the receiver. If a sender component cannot send the data to the receiver component periodically, the receiver can detect a data period failure of the sender. The periodic data failure may be caused by a network failure between the sender and receiver components. The blackboard architecture determines whether it is a network failure or sender's periodic data failure.
- Periodic data processing failure. The data delivered by a robot component to another may need to be processed within the time duration specified. The data

processing time of a component is checked to detect whether a component processes the data within the duration.
- Event processing failure. An urgent event, such as wheel stop, can be communicated between robot components to avoid an accident. Such an event should be processed immediately. A robot component may fail if it does not process the urgent events immediately. An event processing failure is detected if a robot component cannot process a received event as specified.
- Network failure. Distributed robot components cannot communicate with each other if the network encounters a failure. The blackboard architecture may detect the network failures between distributed robot components by cooperating with the operating system
- Actuator failure: It is assumed that an actuator encounters a failure if it cannot commit the commands as specified. Like sensors, it is difficult to detect actuator failures because there are many different types and brands of actuators. But, an interface component to an actuator may know that an actuator encounters a failure if the actuator driver (provided by the actuator supplier) detects the failure. In that case, the interface component notifies the blackboard architecture of the failure.

In order to detect the failures of components, each component has its own profile that describes the runtime conditions in terms of functions provided by components and data processed by components. The component profiles are presented using an Extensible Markup Language (XML).

3.2 Aspect-Oriented Detection of Failures

Aspect-oriented programming modularizes the failure/recovery functions separately from robot application logic. The failure/recovery detection functions need to be plugged in to robot application components in which the detection functions monitor the failure/recovery of the component. As the failure/recovery detection functions are mixed with application functions of components, they can increase the complexity of robot applications. In particular, when the application functions of components are changed, the failure/recovery detection functions can be affected accordingly.

In this paper, the failure/recovery detection functions are designed and implemented as aspects in aspect-oriented programming (AOP). Aspect oriented programming provides robot application developers with a tool to separate concerns from core application functionality. It promotes the separation of each crosscutting concern into its own aspect. An aspect can be broken down into four parts: the aspect, a join point(s), a pointcut(s), and advice. An aspect may be used at various points in applications. For instance, periodic data failure/recovery detection function can be used at multiple points in robot components in order to detect the failures/recovery of periodic data. The periodic data failure/recovery detection is a crosscutting concern of the application separated from the application logic.

An aspect has two important components namely the pointcut(s) and advice. Pointcuts are the moments of execution in an application, commonly referred to as join points. These join points can be object initializations, method calls, and method executions for instance. A pointcut clearly defines the collection of join points at which an aspect should be injected for use. The advice of an aspect provides the additional code or methods required to give the aspect the functionality desired. In this paper, the advice includes the algorithms and methods needed to detect failure/recovery of components. The failure/recovery detection functions have been implemented as advices in the AOP by means of the PostSharp [14] under MSRDS.

3.3 Cause Analysis of Failures

The blackboard architecture analyzes the causes of component failures that are notified by robot components. The blackboard architecture determines which component has failed using the failure messages from components. In case of a periodic data failure, it should keep track of failure messages to find out the failed component(s). A periodic data failure takes place if a component cannot receive a data from sensors or other components periodically. A periodic data can be processed by multiple components in an application.

Once a periodic data failure occurs, multiple components processing the periodic data notify the failures to the blackboard architecture. The blackboard architecture needs to determine which component has caused the failure. Instead of all the components that notify the blackboard architecture of the same failures, the blackboard architecture keeps track of the status of the component(s) that has generated the failure.

The cause of failures is analyzed by means of the dependency relationships between components in terms of data or operations [15]. A component may require data or operations from another. In this case, the two components have a data dependency relationship if a component processes the data coming from another.

The dataflow between components is depicted in (a) of Fig. 2, which is part of the unmanned ground vehicle (UGV) application. The Laser Range Finder (LRF) Interface component and Camera Interface component generate LRF data array and 24 bits raw RGB, respectively. The Car Detection component produces other cars trajectory after analyzing the data that comes from the LRF Interface component and Camera Interface components. Using the data from the Car Detection component, the Dangerous Situation component determines whether it should generate an alarm to prevent a critical accident. The (b) of Fig. 2 depicts the data dependency relationships between components, which are defined based on the dataflow in (a) of Fig. 2. The data dependency relationship between components is represented as the opposite direction of the dataflow between components. The dataflow and data dependency relationship between components are represented with an arrow and a dotted arrow, respectively.

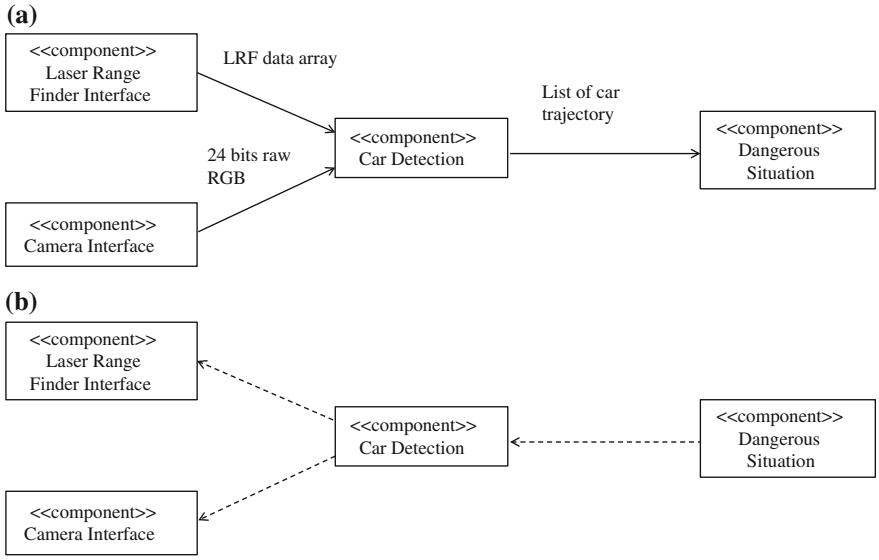


Fig. 2 Dataflow and data dependency relationship between components in UGV application. **a** Dataflow between components. **b** Data dependency between components

Suppose that the Camera Interface component (Fig. 2) is out of service. The failure of Camera Interface component can be detected by the Car Detection component and then by the Dangerous Situation component. Both Car Detection and Dangerous Situation components may notify the blackboard architecture of these failures. The blackboard architecture infers that the Camera Interface component has a failure based on the data dependency relationship.

4 Failure/Recovery Notifier

The failure/recovery notifier in the blackboard architecture informs robot components of either detected or recovered failures. The blackboard architecture can communicate with components via a function call or asynchronous message communication. Each component has failure flags representing the status of different types of failures. The failure flags are set to true or false in response to a notification message. When the value of a failure flag changes, some actions in a component can be taken against the failure message or for the recovery message if the actions have been implemented by robot application developers.

Impact analysis between components determines the scope of notification in the failure/recovery notifier. The impact of a component failure on other components is determined by considering how much a component is affected from a failure. A failure impact may be localized within a component or spread to multiple components in

the application. The impact of a component failure on other components can change depending on the failure types defined in Sect. 3.1.

The failure impact will be modeled by means of an impact level on the dependency relationships between components. The impact level of a component failure on its dependent is categorized as insignificant, tolerable, serious, and catastrophic [15]. An insignificant impact level describes a dependency relationship in which a component uses information from a failed component to verify or increase reliability of component output additionally. A component can provide its full functionality required even though there is no additional information from a failed component with an insignificant impact level.

Figure 3 depicts the impact levels corresponding to the periodic data between components described in Fig. 2, which has one more Camera device for higher reliability. The Laser Range Finder (LRF) Interface component is insignificant for the Car Detection component, which uses the data from the Laser Range Finder (LRF) Interface component and two Camera Interface components to keep track of other cars trajectory. Even though the LRF Interface component fails, the Car Detection component can still produce other cars trajectory using the data coming from the Camera Interface components. The LRF component is added to the application as an additional component so that the Car Detection component can produce high quality of other cars trajectory.

A tolerable impact level describes a dependency relationship in which a component is dependent on the same type of multiple components. The dependent component may have a minor impact if one of the multiple components encounters a failure. However, a tolerable failure does not disrupt the normal functionality of its affected component if some of the multiple components still work. The Car Detection component (Fig. 3) is supported by two Camera components, impact level of each of which is defined as tolerable. Even though one of the Camera Interface components

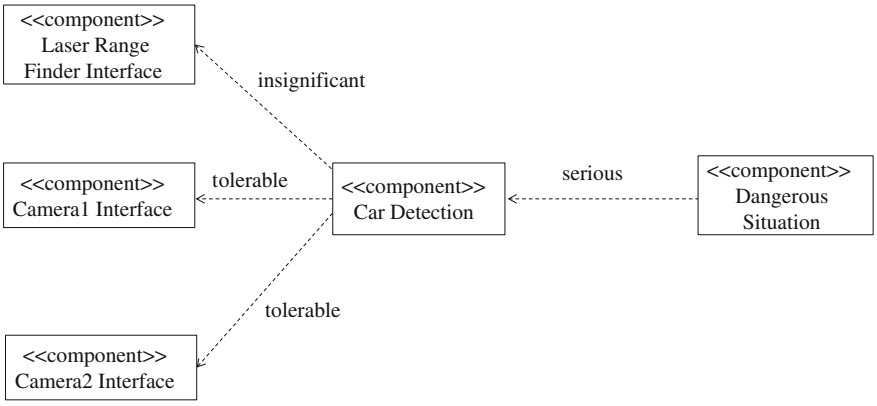


Fig. 3 Impact levels between components in UGV application

fails to capture other cars pictures, the Car Detection component can make other cars trajectory using the remaining Camera Interface component.

The tolerable impact level needs to be changed to the serious impact level if at least a number of tolerable components do not work (referred to as cardinality condition). Each Camera Interface component (Fig. 3) has a tolerable impact on the Car Detection component, but the tolerable impact level changes to serious if both Camera Interface components encounter failures. This is because the Car Detection component requires the data from at least one Camera Interface component so as to generate the reliable output for the Dangerous Situation component.

A component failure that makes serious impact on its dependent component causes the dependent to be out of service. This is because the dependent component cannot provide a reliable output any more without the failed component. Stopping the dependent component could bring a ripple effect to the next dependent components so that it may paralyze part or all of an application. If stopping the dependent components does not lead the application to a total failure, the application can still provide partial services using the remaining components, which have no critical impact from the stopped components. The data generated by the Car Detection component (Fig. 3) is critical to the Dangerous Situation component, so the impact of Car Detection component on the Dangerous Situation component is defined as serious. When the Car Detection component fails, the Dangerous Situation component should stop processing the data.

A catastrophic impact is used to describe a situation where a component failure needs to stop all components constituting an application. An application may encounter a critical accident if all the components do not stop immediately. A catastrophic impact may be associated with safety of robots. The Virtual Robot component in the UGV application controls devices such as engine and steering wheel. A failure of Virtual Robot component may cause the UGV to encounter an accident, which could lead to lose human lives.

5 Blackboard Architecture for MSRDS

5.1 Implementation of Blackboard Architecture with MSRDS

A prototype of the blackboard architecture has been developed for MSRDS applications to validate the proposed approach. In the MSRDS, a robot application is built with multiple components, referred to as services, which are highly decoupled from each other. Figure 4 depicts the blackboard architecture for MSRDS applications. Each service (component) for applications declares its partnerships with the monitoring and notification services in the blackboard architecture. The blackboard architecture communicates with applications' services via message ports.

The monitoring service collects failure information from the application's services. A failure in a service may occur due to the impact of other services' failures.

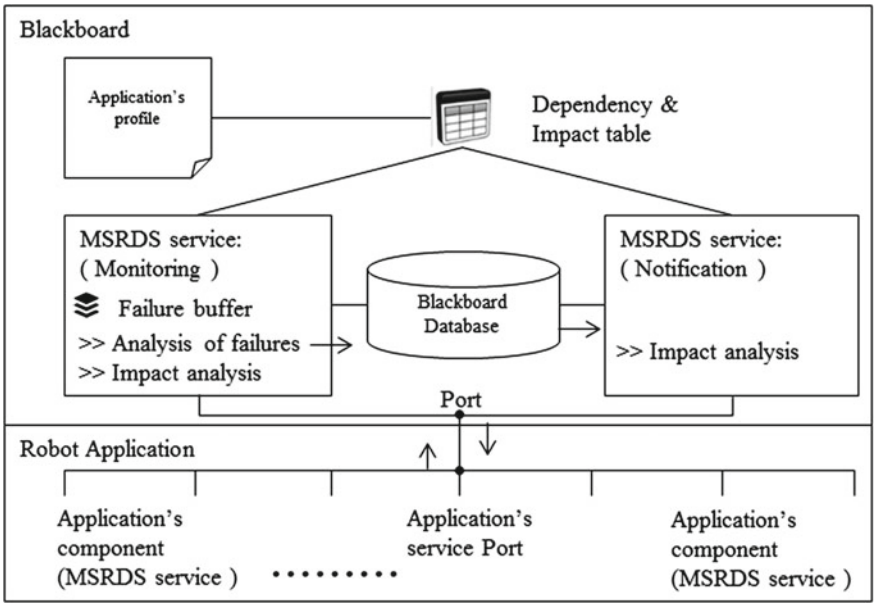


Fig. 4 Blackboard architecture for MSRDS applications

When a failure is reported to the monitoring service, the service stores the failure to a buffer for some period time. The failures in the buffer are analyzed to find out the root cause of the failures using the dependencies between services and failure types. These dependencies are described in application's profile that is represented by an XML. In the application profile, one part defines the dependencies between components and impact levels in their relationship, and the other part includes the cardinality conditions which are explained in Sect. 4. The notification service in the blackboard architecture notifies the components of the failure/recovery of a component. When a new root failure is reported or recovered, the notification service starts the impact analysis to determine the scope of components being notified.

5.2 Unmanned Ground Vehicle (UGV) Application with MSRDS

The unmanned ground vehicle (UGV) application has been developed using the MSRDS simulator in order to validate the blackboard architecture. The UGV drives to the destination place safely without a human driver's intervention. The UGV application is a real-time system requiring high safety in order not to lose human lives from car collapses. The MSRDS provides a three-dimensional simulator in which the services and road environment for the UGV application are constructed.

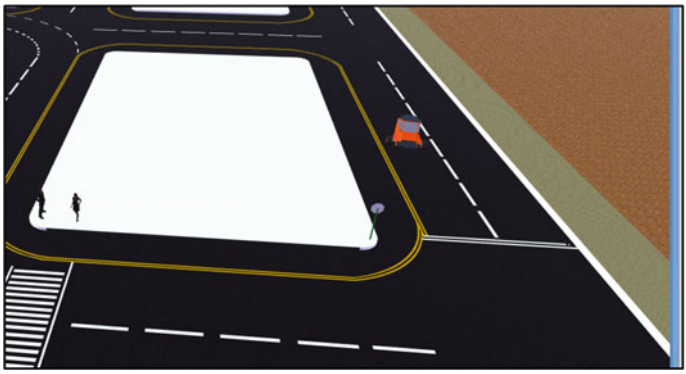


Fig. 5 Unmanned ground vehicle application with MSRDS simulator

Figure 5 depicts a UGV and the road environment that have been developed with the MSRDS simulator. The UGV is equipped with various sensors such as cameras, sonar sensor, laser sensor, and actuators such as wheels, display, speaker, and light. The UGV application is composed of 19 software components (services) including Curb Detection, Dangerous Situation, and Localization.

The blackboard architecture for the MSRDS has been tested with the UGV application in which each service (component) communicates with the blackboard architecture to detect and notify failures/recoveries. For each different type of failures, the application has been designed with intended safety actions. Components can take the safety actions against failures, such as (1) sound UGV's horn, (2) turn on alarm light, (3) slow the speed, (4) drive to side road, and (5) stop the UGV immediately.

5.3 Analysis of Blackboard Architecture

This research has analyzed the performance of applications when the applications run with the blackboard architecture. The blackboard architecture requires additional message communication with application components in terms of monitoring and notification. The messages may affect the performance of applications. For this, this research measures the performances of UGV application with and without the blackboard architecture in order to see how much the blackboard architecture affects the performance of applications.

Three components—Curb Detection, Dangerous Situation, and Localization—in UGV application are selected because these components take important roles in the application. The real execution times for the selected components are measured by recording the time whenever the components start their processes as execution cycles predefined. For each component, 50 samples for the components' execution times

are observed. This experiments are done with 8GB memory, i5-3330(3GHz) CPU and 64 bit Window 8.

The blackboard architecture does not significantly degrade the performance of UGV application under the limited experiments. The performance degradation in execution time varies in the range of 0.03–0.34 % in the Curb Detection, Dangerous Situation, and Localization components. This means that the starting times of each periodic execution cycle of these components have been delayed in the range as the blackboard architecture is deployed with the UGV. However, our other experiments, which measure the UGV application without the blackboard architecture, show that the components have been activated late periodically in the similar range.

6 Related Work

Component-based robot software development [3, 4, 16–19] has become primary concern due to the dramatic increasing of complexity in robot applications. The traditional robot programming has been designed as monolithic. That is, all functions and data in the program are tightly coupled so that developers should know the details to cope with the changes of hardware and software. Component-based software engineering (CBSE) aims to shift a means of system building from traditional approach to component-based approach. Component-based approach reduces the amount of time to develop a new application and makes a system more robust because pre-existing components are already tested.

However, existing robot software framework and platforms, which support development of component-based robot applications, do not provide the runtime failure management approach proposed in this paper. Many researches for component-based robot application development have been done in the field of robotics, providing robot software framework and platforms that help to develop robot applications using robot components or reusable service building blocks. Those approaches include Microsoft Robotics Developer Studio (MSRDS) [5], Robot Operating System (ROS) [6], Robotic Technology Component (RTC) [7], Open Robot Control Software (OROCOS) [8], Middleware for Autonomous Mobile Robots (MIRO) [9, 10], Open Platform for Robotic Service (OPRoS) [12], iRobot AWARE [20], and RT-Middleware [11]. Some of these approaches provide the mechanisms for handling low-level faults, but they do not support the approach that detects runtime failures and notifies the components of them on an application level.

Ad hoc approaches [21, 22] have been proposed for handling runtime failures for robot control software or hardware. Authors in [22] introduce an approach to monitor the driving device of a mobile robot using model-based reasoning. Authors in [21] present a method that detects runtime faults and recovers from the faults in robot control software. This approach covers restricted fault types in client-server based robot control software, whereas our approach handles broader runtime failure types separately from robot applications.

7 Conclusion

This paper has described an approach to detecting and notifying the failures in component-based robot applications using the blackboard architecture. The blackboard architecture provides the detection and notification mechanisms to distributed component-based robot applications in which each component can be monitored and notified of the failures of other components. Robot applications can be developed separately from the blackboard architecture, and then they can be connected to the blackboard in order to get failure detection and notification services. In this paper, the backboard architecture has been implemented for MSRDS applications.

This paragraph describes future research for the blackboard architecture. More MSRDS applications need to be developed to validate the blackboard architecture further. Current blackboard architecture has been tested with the UGV application. Another way of future research is to develop a different version of blackboard architecture for Robot Operating System (ROS) [6], which is another popular tool for robot software development.

Acknowledgments This work was supported in part by the Knowledge Economy Technology Innovation Program of MOTIE/KEIT, Rep. of Korea [10044006, Development of Open Robot Middleware Supporting User-Friend Developer Tools and Standard Robot API Components]

References

1. Gomaa, H.: Designing Concurrent, Distributed, and Real-Time Applications with UML. Addison-Wesley, Boston (2000)
2. Pfleeger, S.L., Atlee, J.M.: Software Engineering Theory and Practice, 3rd edn. Prentice-Hall, Upper Saddle River (2006)
3. Crnkovic, I.: Component-based Approach for Embedded Systems. IEEE Press, New York (2004)
4. Brugali, D., Scandurra, P.: Component-based robotic engineering. Robot Autom Mag IEEE **16**(4), 84–96 (2009)
5. Jackson, J.: Microsoft Robotics Studio: A Technical Introduction, IEEE Robotics and Automation Magazine, Dec, pp. 82–87 (2007)
6. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an open-source robot operating System. In: ICRA Workshop on Open Source Software, Kobe Japan (2009)
7. OMG The Robot Technology Component Specification. http://www.omg.org/spec/ (2012)
8. Soetens, P.: The OROCOS (Open Robot Control Software) Component Builder's Manual. Version 1.10.2. FMTC (2007)
9. Enderle, S., Utz, H., Sablatnog, S., Simon, S., Kraetzschmar, G., Palm, G.: MIRO: middleware for autonomous mobile robots. In: Telematics Applications in Automation and Robotics (2001)
10. Utz, H., Sablatnog, S., Enderle, S., et al.: Miro-middleware for mobile robot application. In: IEEE Transactions on Robotics and Automation, pp. 493–497 (2002)
11. Ando, N., Suehiro, T., Kitagaki, K.,et al.: RT-Middleware: distributed component middleware for RT (robot technology). In: IEEE/RSJ International conference on robots and intelligent systems 2005, pp. 3555–3560 (2005)

12. Song, B., Jung, S., Jang, C., Kim, S.: An introduction to robot component model for OPRoS (Opend Platform for Robotic Services). In: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Venice, Italy (2008)
13. Jang, C., Lee, S., Jung, S., Song, B., Kim, R., Kim, S., Lee, C.: OPRoS: a new component-based robot software platform. ETRI J **32**(5), 646–656 (2010)
14. PostSharp 3.1 http://doc.postsharp.net/conceptual-documentation (2013)
15. Shin, M. E., Kang, T. Kim, S. Jung, S., Roh, M.: Reconfiguration of robot applications using data dependency and impact analysis. In: 24nd International Conference on Software Engineering and Knowledge Engineering, San Francisco, July 1–3, pp. 684–687 (2012)
16. Wei, H. Duan, X., Li. S., Tong, G., Wang, T.: A component based design framework for robot software architecture. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS09), St. Louis, USA, Oct 11–15, pp. 3429–3434 (2009)
17. Åkerholm, M., Möller, A., Hansson, H., Nolin, M.: Towards a dependable component technology for embedded system applications. In: 10th IEEE International Workshop on Object-oriented Real-Time Dependable Systems (WORDS05), Feb, Sedona, Arizona, USA (2005)
18. Biggs, G.: Flexible, adaptable utility components for component-based robot software. In: 2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, May 3–8 (2010)
19. Brooks, A., Kaupp, T., Makarenko, A., Oreback, A., Williams, S.: Towards component-based robotics. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS05), Alberta, Canada, 2005, pp. 163–168 (2005)
20. IRobot Create http://www.irobot.com (2011)
21. Steinbauer, G., Morth, M., Wotawa, F.: Real-time diagnosis and repair of faults of robot control software. In: Bredenfeld, A., et al. (eds.) RobotCup 2005, LNAI 4020, pp. 13–23 (2006)
22. Hofbaur, M., Kob, J., Steinbauer, G., Wotawa, F.: Improving robustness of mobile robots using model-based reasoning. J. Intell. Robot Syst. **48**(1), 37–54 (2007)

Coordination of Modular Robots by Means of Topology Discovery and Leader Election: Improvement of the Locomotion Case

José Baca, Bradley Woosley, Prithviraj Dasgupta, Ayan Dutta and Carl Nelson

Abstract An important aspect of successful locomotion in Modular Self-reconfigurable Robots (MSRs) is to be able to autonomously coordinate the movement of the modules so that the robot can move towards the goal. We consider the locomotion problem in a partially distributed setting where multiple MSRs (disconnected groups of connected modules) are within the communication range of each other and modules do not have a priori information about other modules that belong to the same configuration. Coordinating the movement of modules in such a setting becomes a challenging problem because of the limited perception and computation resources available on each module. To address these problems, we propose a strategy that first combines neighbor-to-neighbor message passing techniques via infrared and wireless communication to enable each module to autonomously determine the set of modules that belong to the same MSR. The strategy then uses a distributed leader election algorithm to identify the leader, which thereafter coordinates the actions of the modules in its configuration. We have verified the performance of our approach using an accurately simulated model of the ModRED MSR within the Webots simulator and in the embedded system of ModRED (This work was done as part of the ModRED project which is supported by NASA EPSCoR grant no. NNH11ZHA003C.). It is shown that our strategy can successfully determine the set of connected modules, elect a leader for each configuration and coordinate the locomotion of MSRs for different numbers of modules.

Keywords Modular robots · Coordination · Locomotion

J. Baca (✉) · B. Woosley · P. Dasgupta · A. Dutta
Computer Science Department, University of Nebraska at Omaha, Omaha, USA
e-mail: jbacagarcia@unomaha.edu

P. Dasgupta
e-mail: pdasgupta@unomaha.edu

A. Dutta
e-mail: adutta@unomaha.edu

C. Nelson
Mechanical and Materials Engineering Department, University
of Nebraska-Lincoln, Lincoln, USA
e-mail: cnelson5@unl.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_31

1 Introduction

Modular self-reconfigurable robots (MSRs) are robotic systems consisting of modules which can connect with each other to form different shapes. Because of their highly dexterous nature, MSRs are attractive for maneuvering and performing tasks autonomously in environments that are highly unstructured [1]. To enable a MSR to maneuver successfully, it is essential to ensure that each module performs its required movement in coordination with other modules, so that the overall desired motion of the MSR is manifested. This problem is called the MSR locomotion problem; it becomes challenging, especially as the size of the MSR grows, because of the limited perception capabilities and computation resources on each MSR module, and the time delay between the motion of modules that are not adjacent to each other in the MSR.

To address this problem, the motion between the different modules needs to be coordinated so that each module is provided information about its appropriate movement at each time step [2–5]. Some techniques to solve this problem include using a centralized pattern generator [6] that generates appropriate time-delayed signals to actuate each module, generating a synchronization signal successively on each module that can be perceived by adjacent modules to commence actuation [7], or defining a module's role as a function of time [8]. For instance, the hormone-based implementation offers two advantages: it keeps the modules of the system synchronized and modules can be added or removed from the chain configuration. The modules stay synchronized because upon starting an action, the module at the head of the chain passes a message to the following module (via IR) instructing which action it should perform in the current step. Similarly, the second module propagates the message to the following module instructing which action it should perform. The process continues until it reaches the end of the chain. Modules can be added to the chain configuration because the message is given by the predecessor of each new module. Two major drawbacks of this approach are observed. First, the initiator has to be defined by the user or programmer, and second, if one of the modules in the chain configuration fails (e.g., hardware malfunction, etc.) then locomotion cannot be accomplished because the propagation of the message has been broken, as shown in Fig. 2a. Another interesting technique is role-based control. This approach does not use a gait control table with a sequence of actions; rather, the positions of the internal joints of a module are defined as a function of time. The technique presents three main features: the speed can be changed simply by choosing different periods in the function, it is not synchronization-based, and it does not require an initiator. One drawback is that since the function is based on the module's internal representation of time, any differences in clock speeds (within the embedded system) between modules can lead to irregular motions of the configuration.

In this paper, we have considered interesting features from both these techniques and fused them—we consider a scenario where multiple MSRs (disconnected groups of connected modules) are in close proximity of each other; each module has to autonomously determine the set of modules within the configuration it belongs to;

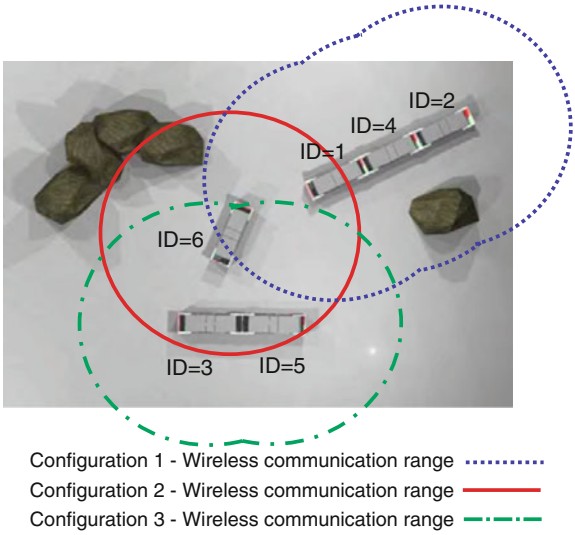


Fig. 1 In a typical scenario, it is possible to find two or three configurations (made up of n modules) working in the same area. The assignment of a leader's role to one module in each configuration can facilitate the coordination of the entire configuration and help achieve a common goal. Since each configuration can have different tasks or goals, it is convenient that the leader is elected by, and only by, the modules comprising the configuration

and finally each MSR has to select a module, called the leader, that coordinates all the modules in the same MSR to move in the desired direction. One of the advantages of having a leader in the configuration is that if during a mission a new task has to be assigned to that specific configuration then it is easier to establish connection between the leader and the control station and transmit the new instructions. We focus on finding a way to coordinate the activities performed by each of the modules to accomplish a common goal (for instance, the activation of a goal-driven controller presented in [9]). At the same time, we develop approaches that meet two main criteria, i.e., the limited resources that can be found in different modular robotic systems (e.g., limitations of sensors, computation capability, size, etc.) and the approach has to consider an equilibrium between flexibility, time response and complexity for a small set of modules (due to payload constraints in space missions). The approach assigns the leader's role to a module belonging to the same configuration. Assuming that all modules are identical (as in a homogeneous system), it will not matter which module becomes the leader. However, it is important that the leader is automatically elected only by the modules forming the same configuration (Fig. 1). The election algorithm should be running only in the corresponding modules. For such purpose, we propose that each module maps the topology's configuration to know which modules are involved in the election process. In other words, the map of the modular configuration (MMC) generated by each module contains the modules that are currently in the same MSR and their position in the configuration (Fig. 2)

The work in this paper also complements and improves our previous work in MSR locomotion where we had proposed a fuzzy logic controller (FLC) that prescribes a movement for each module in an MSR connected in a chain configuration without coordination [9]. By adding coordination into the system, we are able to improve the

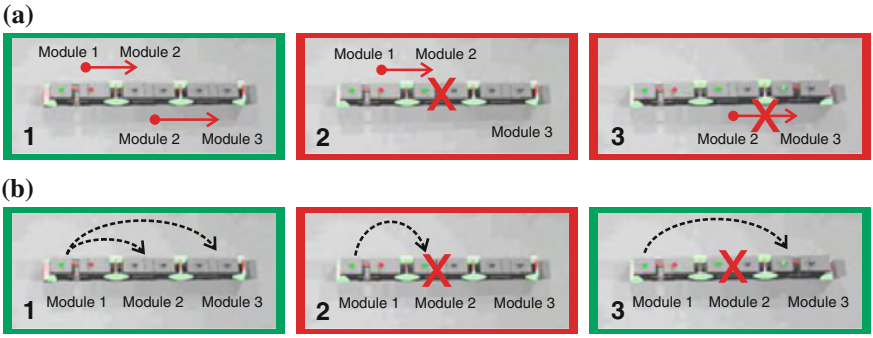


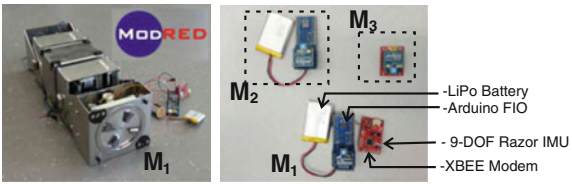
Fig. 2 **a** The propagation of the message (via IR) is broken if one of the modules between the chain configuration fails. **b** The advantage of having point-to-point communication is that even if a module fails in the configuration, actions can be executed by other modules at the corresponding time

overall performance of the MSR when performing locomotion. We have verified the performance of our algorithm using an accurately simulated model of the ModRED MSR [10] within the Webots simulator while using multiple MSRs in chain configurations, with different numbers of modules on each MSR. Moreover, we have implemented the strategy in the hardware of ModRED to verify its real performance. Our results show that our algorithm can successfully determine the set of connected modules, elect a leader for each MSR and coordinate the locomotion of MSRs to move them in the goal directio.

2 ModRED System

ModRED (*Modular Robot for Exploration and Discovery*) is a homogeneous modular robot system that is suitable for efficient maneuver over unstructured surfaces such as extra-terrestrial environments [11]. As an autonomous system, modular robots must be capable of sensing their environment and acting on this information for task completion purposes. Each of the ModRED modules is equipped with necessary electronics to give them such autonomy, as shown in Fig. 3. Each module performs the computation and control tasks using two Arduino Fio (ATmega328P)

Fig. 3 Embedded system of a ModRED module



microcontrollers or one BeagleBone Black board. For powering the overall system including all the sensors, actuators and microcontrollers, a rechargeable lithium-polymer battery pack is used. Each module is equipped with an XBee modem directly connected to the Arduino board to enable wireless communication among the modules. In addition to this, an array of infrared (IR) sensors is used for communication, proximity sensing and local localization strategies. For obstacle detection purposes, and to ensure successful docking, bump switches are incorporated in the front/rear faces of the docking brackets.

3 Topology Discovery

The idea of mapping the configuration's topology or location of each of the modules within a modular configuration is to give to the system knowledge of the arrangement of the various modules in the actual system, as in network topologies. In other words, each module would know the neighbors of each of the modules in the current configuration. This feature would allow the execution of complex tasks by specifically coordinating each module comprising the current configuration. For instance, in this work we will coordinate the activation of a fuzzy controller contained in each of the modules at different intervals of time so that the overall locomotion's efficiency of the global configuration increases.

3.1 Mapping Local Neighbors

Infrared (IR) communication is used to allow the modules to exchange their identification number (ID) with the neighbors connected to either the front or rear connecting faces, as shown in Fig. 4a. All modules are given unique IDs which allow them to be distinguished from all the other modules in the area and communicate wirelessly among them. The IR sensors are set up in such a way that communication is only

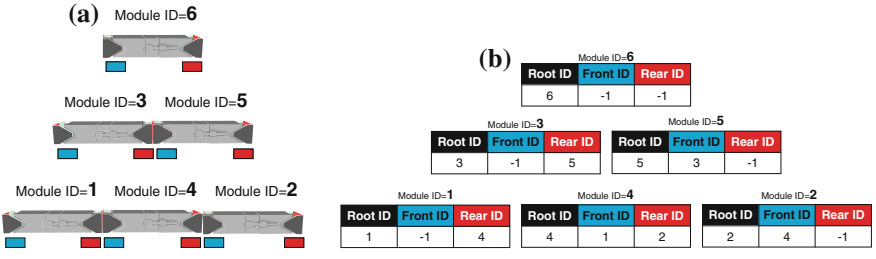


Fig. 4 **a** Example of neighbors connected to the front and rear connecting faces. **b** Local configuration structure (LCS) generated by each module. The value −1 means no module attached to the connecting face

possible if a module is connected to the connecting face. Each module transmits its ID out both the front and rear connecting faces, and listens for an ID to be received on the front and rear data channel. When the data has been collected, it is placed into a *local configuration structure* (Root ID | Front ID | Rear ID), that shows the current module's ID, and the ID of the module(s) connected to its front and rear connecting faces, as shown in Fig. 4b.

3.2 *Broadcasting My Neighbors' List Among the Modules*

Wireless communication is used by each of the modules to broadcast its previously mentioned *local configuration structure* (LCS) and its current clock value (it is used in Sect. 4 for the leader election process) to all modules within wireless communication range. It is important to mention that when broadcasting the data to the entire set of modules, there might be modules belonging to a different configuration but still within the communication range. For instance, Fig. 1 shows three different configurations (a singleton, two module and three module configurations) displaying their wireless communication range. At this point, each module does not know to which configuration it belongs, but it broadcasts and receives the data to/from the modules within the communication range. For illustration purposes, Fig. 5 shows the received LCSs for modules with ID = 5, 6 and 4.

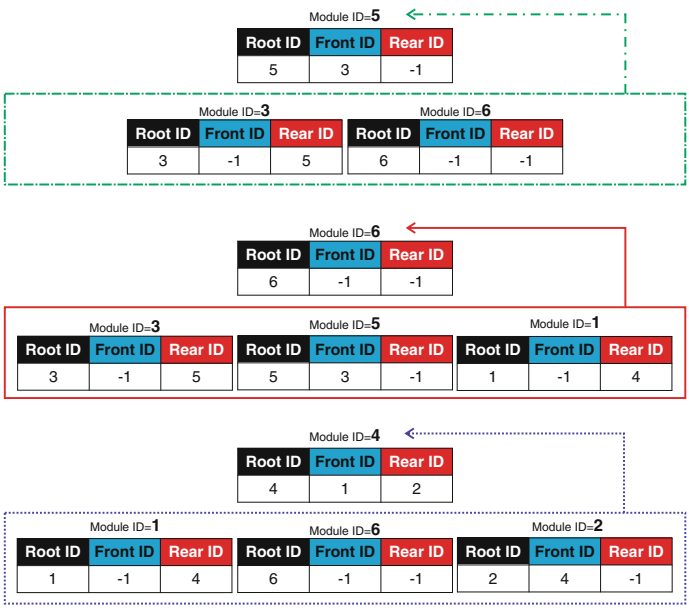


Fig. 5 Received LCSs from modules within communication range (obtained from Fig. 1)

3.3 Topology Generation Process

In order to generate the final configuration's topology, each module has to analyze and merge its own LCS and the received LCSs into a single structure. For example, Fig. 1 shows a scenario with three different configurations. Configuration 1 is formed by three modules with ID = 1, 4 and 2. Due to the LCS generated by IR communication, module ID = 4 knows that module ID = 1 and module ID = 2 are connected to its front and rear connecting faces respectively, as shown in Fig. 5 (bottom case). However, it doesn't know if there are other modules connected to the same configuration beyond those two modules. Then, through wireless communication, it receives the LCSs from modules with ID = 1, 2 and 6. All LCSs received may or may not belong to the same configuration. To find out the configuration's topology, Algorithm 1 is used. By looking at each LCS from each ID and the correct arrangement of data (line 16), each of the modules finds out that module ID = 6 does not belong to configuration 1. The final map is displayed as |1|4|2| or |2|4|1|.

Algorithm 1: Building topology's configuration structure

Input: n: size of the set of modules, LCS_i: local configuration structure of module i, ID: module's id
Output: MMC: map of the modular configuration

```
1  creation of a structure of 2n size ≡ MMC'
2  currentID = ID; index = n
3  while currentID ≠ −1 and currentID ∋ MMC' do
4      store currentID in MMC'_index
5      i = currentID from LCS_i, currentID = FrontID
6      if currentID = −1 or currentID ∈ MMC' then
7          from LCS_i, currentID = RearID
8      index++
9  from LCS_ID, currentID = RearID
10 while currentID ≠ −1 and currentID ∋ MMC' do
11     store currentID in MMC'_index
12     i = currentID from LCS_i, currentID = FrontID
13     if currentID = −1 or currentID ∈ MMC' then
14         from LCS_i, currentID = RearID;
15     index−−
16 ∃j ∈ {0, |MMC'| − 1}∀i ∈ {0, |MMC| − 1}MMC_i ← MMC'_j, if MMC'_j ≠ NULL
```

4 Leader Election

The coordination of modules would be led by a module who has been elected from among the modules belonging exclusively to the same configuration. The advantage of providing the "map" of the configuration to each of the modules is that any module

is eligible to play the leader's role. Although it is possible to set the module that is located at the front of the configuration as a leader (user-defined), we opted for a leader selection based on an election process that provides greater robustness to the system. This feature allows the system to automatically elect a new leader in case of the leader's malfunction. Since each ModRED module contains its own microprocessor, hence, its own timer, we decided to use the clock's value as an input variable for the election algorithm. To continue with the philosophy of implementing light-weight processes for the limited resources that can be found in embedded systems, we have chosen a modified version of the bully algorithm [12] which satisfies the criteria.

4.1 Bully Algorithm

The leader's role would be assigned to the module with the highest clock value at the beginning of the election process. When the election starts, the clock value is stored and transmitted to all modules within the communication range. This stored clock value will remain the same throughout the entire election process, and is only changed when a re-election is requested. Throughout this time, the module's clock continues normally and is never held constant. Only modules in the current map of the modular configuration (MMC) are eligible to become the leader. Each configuration may have its own task, and leaders need to stay in constant communication with those in their configuration. The list of modules inside the current configuration comes from the MMC presented in Algorithm 1. Once the modules know the clock values for every module in the configuration, the clock values are compared to determine who the leader is. The leader then notifies each module in the MMC that it is the leader. For the current leader to remain as the leader, it must transmit a keep-alive packet periodically.

A propagation function (user defined) provides the order the modules in the configuration should be activated to provide the required motion. For demonstration purposes, a simple propagation function has been implemented (Eq. 1). The leader activates each module starting at one end of the chain configuration and each time step activates the next module in the chain until it reaches the other end of the configuration. At this point, the propagation function is reset to the start of the configuration. The following equation describes the order in which modules are activated.

$$module = MMC_{[time\%size(MMC)]} \tag{1}$$

where % is the modulo operator, size() returns the number of modules in MMC and MMC[i] returns the ID of the module in the ith position of MMC from the end of the configuration.

5 Experimental Results

In this section, we present various experimental results from applying this technique in an accurately simulated model of ModRED within the Webots robot simulator and its implementation in the embedded system of ModRED. We have tested the approach in four different scenarios.

5.1 Simulation Results

Each scenario displays situations in which there are two and three configurations and each configuration is made of one up to four modules, as shown in Fig. 6a–d. The configurations are placed in the scenario so that all modules are within wireless communication range. Figure 6e shows the results of the election process. Each configuration is represented by a geometric shape and each case is shown in a different

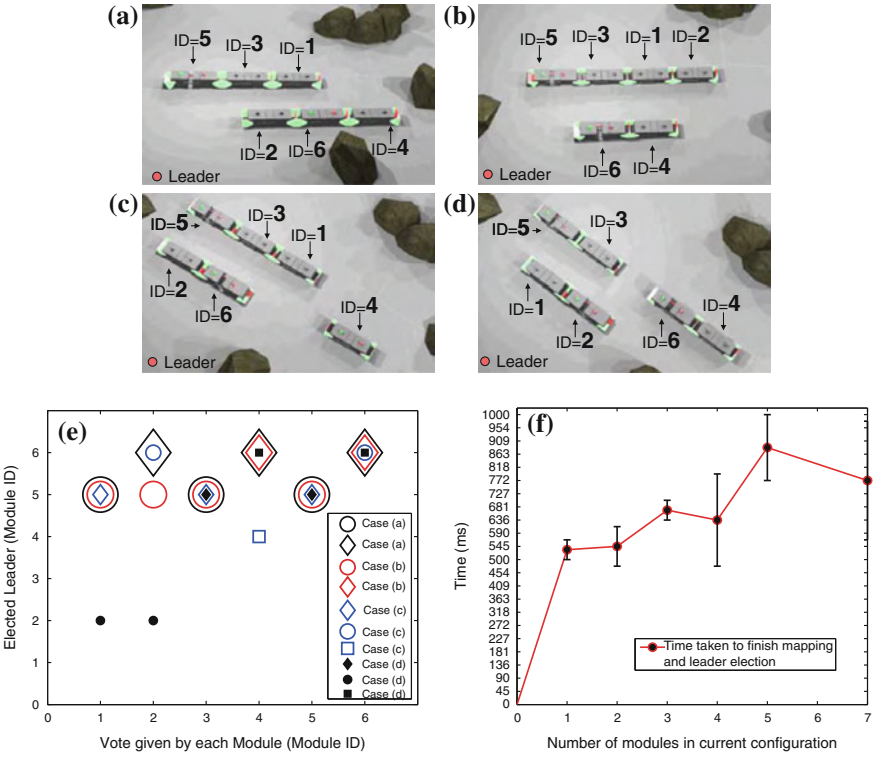


Fig. 6 **a–d** Cases with different configurations. **e** Results of the election. **f** Time taken to complete the election. Set of five runs in each case

color. It is observed that all modules belonging to the same configuration agree with the elected leader. For example in case (b), four modules belonging to the same configuration (red circle) agree that module with ID = 5 becomes the leader. Similarly, the modules comprising the second configuration (red diamond) agree that module with ID = 6 becomes the leader. Figure 6f illustrates the time taken (milliseconds) to finish mapping the topology of the configuration and the corresponding leader. Once the leader's role has been assigned to the configuration, the leader coordinates the activation of modules' actions using the propagation function (Eq. 1). This function is used to send the activation's signal to each of the modules in different ways.

Additionally, we have implemented this coordination strategy along with our previous work (a distributed fuzzy controller) to demonstrate how the coordination affects the behavior of the configurations. The experiments show that when applying the coordination approach along with the same controller's output to move in one direction, it is possible to change the direction of the configuration's displacement. It is also shown that when implementing the coordination strategy in longer configurations, it allows the correct displacement of the configuration. Videos of the experiments are given as an attachment and additional information can be found at: http://cmantic.unomaha.edu/projects/modred/index.htm.

5.2 *Implementation in the Hardware of ModRED*

We have checked the performance of the strategy by implementing the algorithms in the embedded system of ModRED and performed an experiment, as shown in Fig. 7. The objective is to verify if the strategy is able to discover the topology of the configuration of a randomly selected module and then to elect the appropriate leader according to the clock values. The experiment consists of two MSRs within

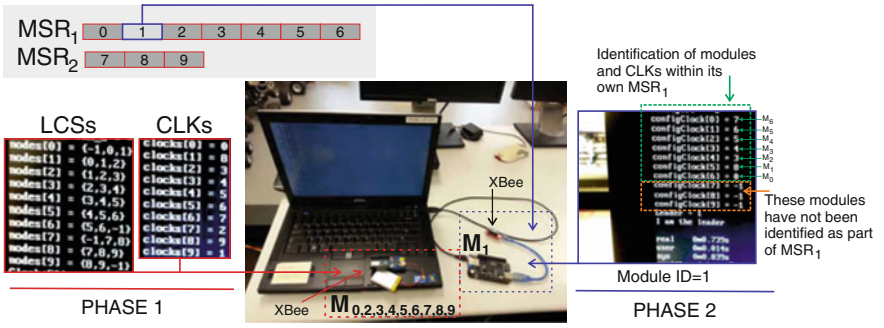


Fig. 7 Implementation and verification of the strategy in the embedded system of ModRED. In phase 1, all modules generate the local configuration structures (LCSs) with the corresponding clock values (CLKs) and broadcast them. In phase 2, each module identifies the modules that are in the same configuration and determines if it is the leader

the communication range with different numbers of connected modules. MSR_1 is made of 7 modules (Module ID = 0, 1, 2, 3, 4, 5, and 6) and MSR_2 is formed by 3 modules (Module ID = 7, 8, and 9). The module with ID = 1 (M_1) in the experiment represents a module in an initially unknown chain configuration (BeagleBone+XBee enclosed by a blue square). The Arduino+XBee board (enclosed by a red square) represents all the other modules in the environment by transmitting each of the LCSs from that set of modules that would be broadcasting in the real case of having 9 modules (BeagleBone+XBee boards). Phase 1 of the experiment starts after all the individual LCSs have been generated. The Arduino broadcasts a series of data packets, each packet containing the LCS and clock value of one of the modules. M_1 transmits its own LCS/clock combination while listening for the LCSs from the other modules. Once it has received the same number of LCSs as there are modules in the environment, or a timeout has expired between receiving new LCSs, the data is passed to the next phase. In phase 2, the algorithm determines which modules are in the same MSR of M_1, and determines the leader of this specific configuration. The solution is enclosed by a green dotted box. The entire process (phase 1 and phase 2) takes approximately 740 ms to find the solution. The size of the algorithm is 65 kB, and it takes 3 kB of RAM.

6 Conclusions and Future Work

In this work we have presented a coordination strategy by which the modules comprising a configuration agree on the election of a leader. This leader is elected only by the modules belonging to the same configuration. The technique first builds a map of the modular configuration (MMC) based on the combination of local configurations structures (LCSs) shared by each module within communication range. Then, modules in the MMC select a leader by introducing clock values into a bully algorithm. The leader makes use of a propagation function to coordinate the activation of actions inside each of the modules belonging to the same configuration. The main contribution of this work lies in the design of having a distributed leader election among the modules in the configuration, which brings robustness to the system in case of the leader's failure and the possibility of direct control on the activation of each module's actions without chain-propagation restrictions. In addition, by running the propagation function only on the leader module, we avoid problems that arise from modules having different clock speeds.

We have shown our approach works successfully on a ModRED model inside the Webots simulator and in the real hardware of ModRED. The experimental results demonstrate that the strategy gives the correct information to the system, allowing the correct election of a leader. In addition, we have implemented the coordination strategy along with our previous work (distributed fuzzy controller for locomotion) and performed some locomotion experiments. It can been seen how the coordination technique changes the overall behavior of the configuration using the same controller's output inside each module. We have tested different configuration sizes to

get their performance. We continue working with a small set of modules since we try to keep an equilibrium between flexibility and complexity of the system. The strategy has been designed considering actual sensors, devices, and microprocessors found in ModRED modules. A good advantage of this type of approach is that it can run in limited embedded systems such as in ModRED modules. As future work, we plan to extend the coordination strategy to more complex configurations such as lattice configurations by substituting the array of LCSs with a matrix of LCSs and perform communication experiments [13].

References

1. 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)
2. Butler, Z., Fitch, R., Rus, D.: Distributed control for unit-compressible robots: goal-recognition, locomotion, and splitting. IEEE/ASME Trans. Mechatron. **7**(4), 418–430 (2002)
3. Christensen, D., Schultz, U., Stoy, K.: A distributed strategy for gait adaptation in modular robots. In: IEEE International Conference on Robotics and Automation, May 2010, pp. 2765–2770
4. Yu, C.-H., Werfel, J., Nagpal, R.: Coordinating collective locomotion in an amorphous modular robot. In: IEEE International Conference on Robotics and Automation (2010)
5. Baca, J., Rossi, C., Ferre, M., Aracil, R.: Cooperative task execution between modular robots based on tight-loose cooperation strategies. In: Proceedings of IEEE International Conference on Robotics and Automation (2011)
6. Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Murata, S., Kokaji, S.: Automatic locomotion pattern generation for modular robots. In: IEEE International Conference on Robotics and Automation, pp. 714–720 (2003)
7. Shen, W.M., Salemi, B., Will, P.: Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. IEEE Trans. Robot. Autom. **18**(5), 700–712 (2002)
8. Stoy, K., Shen, W.-M., Will, P.: Global locomotion from local interaction in self-reconfigurable robots. In: 7th International Conference on Intelligent Autonomous Systems, pp. 309–316 (2002)
9. Baca, J., Dasgupta, P., Hossain, S., Nelson, C.: Modular robot locomotion based on a distributed fuzzy controller: the combination of ModRED's basic module motions. In: International Conference on Intelligent Robots and Systems (2013)
10. Baca, J., Hossain, S., Dasgupta, P., Nelson, P. C., Dutta, A.: ModRED: hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. Robot. Auton. Syst. **62**(7), 1002–1015 (2013)
11. Hossain, S.G.M., Nelson, C.A., Dasgupta, P.: Hardware design and testing of ModRED—a modular self-reconfigurable robot system. In: Proceedings of ASME/IEEE International Conference on Reconfigurable Mechanisms and Robots (2012)
12. Garcia-Molina, H.: Elections in a distributed computing system. IEEE Trans. Comput. **31**, 48–59 (1982)
13. Fitch, R., Lal, R.: Experiments with a ZigBee wireless communication system for self-reconfiguring modular robots. In: Proceedings of IEEE International Conference on Robotics and Automation (2009)

Muscle Synergy Analysis of Human Standing-up Motion Using Forward Dynamic Simulation with Four Body Segment Model

Qi An, Yuki Ishikawa, Tetsuro Funato, Shinya Aoi, Hiroyuki Oka, Hiroshi Yamakawa, Atsushi Yamashita and Hajime Asama

Abstract Human motor behavior can be generated by distributed system. In this study, human standing-up motion is focused as an important daily activity. Especially, 13 muscle activation of lower body and trunk measured during human standing-up motion is decomposed into small numbers of modules of synchronized muscle activation called muscle synergy. Moreover human musculoskeletal model is developed with four rigid body segments based on dynamics and anatomical characteristics of human body. Forward dynamic simulation with the developed model showed that four muscle synergies had their own contribution toward body function: bending forward, moving the center of mass forward, extending whole body, and decelerating the center of mass. Results also indicated that combinations of four modules of synchronized muscle activation could generate human standing-up motion rather than controlling individual muscles.

Keywords Muscle synergy · Human standing-up motion · Musculoskeletal model

1 Introduction

In this study, we analyze the human motion in terms of distributed modules of synchronized muscle activation called muscle synergy. When humans move, they need to incorporate with their redundant body system, i.e. humans have to control larger degrees of freedom of muscles than those of joints in order to achieve the targeted kinematics. Therefore, muscle activation cannot be determined even if the target kinematics is given. To solve this ill-posed problem, the concept of muscle synergy

Q. An (✉) · Y. Ishikawa · H. Oka · H. Yamakawa · A. Yamashita · H. Asama
The University of Tokyo, 3-7-1 Hongo, Bunkyou-ku, Tokyo 1138656, Japan
e-mail: anqi@robot.t.u-tokyo.ac.jp

T. Funato
The University of Electro-Communications, 1-5-1 Chofugaoka,
Chofu, Tokyo 1828585, Japan

S. Aoi
Kyoto University, Kyoto Daigaku Katsura, Nishikyo-ku, Kyoto 6158540, Japan

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_32

was previously proposed by Bernstein [1]. It suggested that humans did not control their individual muscles, but they autonomously coordinated muscle synergies. Regarding the analysis of muscle synergy, it has been suggested that five modules of synchronized muscle activation can account for a large amount of muscle activities during human locomotion [2]. Another study demonstrated that some basic behavior of a frog movement could be explained with small sets of synergies [3].

In our previous study, we analyzed human standing-up motion as an important daily activity, where we developed a musculoskeletal model to ensure that three muscle synergies were corresponded to characteristic kinematic movement of human standing-up motion [4]. Moreover forward dynamic simulation validated that three muscle synergies could generate human standing-up motion. However, our previous study considered only the muscles in lower body and it could not explain one of the four important phases of the standing-up motion, which was forward bending of trunk. In order to fully understand and express human standing-up motion with muscle synergies, it is necessary to extract important modules from both lower body and upper trunk.

Therefore, in this study, our objectives are to elucidate important muscle synergies from measured muscle activation of both lower body and trunk of human standing-up motion. Moreover, musculoskeletal model is developed to represent human body and muscles of lower body and trunk to clarify that human standing-up motion can be achieved by four muscle synergies.

2 Methods

2.1 Synergy Model

This paper has employed the muscle synergy model to represent muscle activation of human movement. It assumes that muscle activation during human movement can be decomposed into spatial structure and temporal structure. Spatial structure is defined as muscle synergy, and it determines relative excitation level of muscles. On the other hand, temporal structure is defined as weighting coefficient which determines time-varying amplitudes of muscle synergies. This muscle synergy model is expressed by

$$\mathbf{M} \cong \mathbf{WC}, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{n \times T_{\max}}$ is a muscle activation matrix expressed as in

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_1(t) \\ \mathbf{m}_2(t) \\ \vdots \\ \mathbf{m}_n(t) \end{pmatrix} = \begin{pmatrix} m_1(1) & \cdots & m_1(T_{\max}) \\ \vdots & \ddots & \vdots \\ m_n(1) & \cdots & m_n(T_{\max}) \end{pmatrix}. \tag{2}$$

Matrix $\mathbf{M}$ consists of muscle activation vector $\mathbf{m}_{i(i=1,\ldots,n)}$ to indicate discrete time-varying muscle activation for different n muscles. Its element $m_i(t)$ indicates muscle activation level of ith muscle at time t ($1 \leq t \leq T_{\max}$). $\mathbf{W} \in \mathbb{R}^{n \times N}$ and $\mathbf{C} \in \mathbb{R}^{N \times T_{\max}}$ represent muscle synergy and weighting coefficient matrices respectively by

$$\mathbf{W} = (\mathbf{w}_1 \cdots \mathbf{w}_N) = \begin{pmatrix} w_{11} & \cdots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nN} \end{pmatrix}, \tag{3}$$

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_1(t) \\ \mathbf{c}_2(t) \\ \vdots \\ \mathbf{c}_N(t) \end{pmatrix} = \begin{pmatrix} c_1(1) & \cdots & c_1(T_{\max}) \\ \vdots & \ddots & \vdots \\ c_N(1) & \cdots & c_N(T_{\max}) \end{pmatrix}. \tag{4}$$

Muscle synergy matrix $\mathbf{W}$ consists of muscle synergy vector $\mathbf{w}_j$ to represent jth muscle synergy ($j = 1, \ldots, N$). Elements of the muscle vector $\mathbf{w}_j$ is w_{ij} to indicate activation level of ith muscle in jth muscle synergy. Time-varying weighting coefficient matrix $\mathbf{C}$ consists of the vector $\mathbf{c}_j$, and its component $c_j(t)$ indicates weighting coefficient of jth synergy at time t.

Figure 1 shows a schematic design of the muscle synergy model. In the figure, three muscle synergies represent muscle activation. Figure 1a indicates three muscle synergies ($\mathbf{w}_{1,2,3}$) and the bars in each square show excitation level of n muscles involved in jth muscle synergy. On the other hand, Fig. 1b shows the corresponding weighting coefficients ($\mathbf{c}_{1,2,3}(t)$). In Fig. 1c, n muscle activations are expressed by linear summation of three muscle synergies and their weighting coefficients. In order to decide muscle synergy matrix $\mathbf{W}$ and its time-varying weighting coefficients $\mathbf{C}$, non-negative matrix factorization (NNMF) [5] is used.

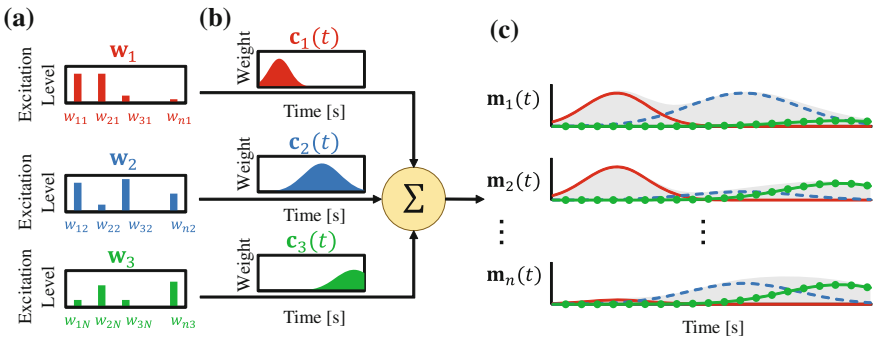


Fig. 1 Muscle Synergy Model. **a** Muscle synergy determines relative excitation level of each muscle. **b** shows time-varying weighting coefficients corresponding to synergies. **c** shows muscle activations. *Red solid lines*, *blue dashed lines*, and *green solid lines* with *circle* markers show muscle activation generated from each muscle synergy and corresponding time-varying weighting coefficients. It indicates that three muscle synergies and their weighting coefficients generate n muscle activations in this example. **a** Spatial structure. **b** Temporal structure. **c** Muscle activation

It is necessary to decide the best number of muscle synergies to represent human standing-up motion. Coefficient of determination R^2 is used to evaluate how much variance of observed muscle activations can be explained from muscle synergies. The number of muscle synergies is decided from two criteria. The first criterion is that muscle synergies can account for a large amount (>95 %) of total variance of muscle activation [6]. The other criterion is how the additional synergy affects the performance of muscle synergy to explain the variance of muscle activation. One-factor analysis of variance (ANOVA) is used to assess the effect of the number of muscle synergies on the accuracy of the model. If there is a statistical significance, the post-hoc test (Tukey-Kramer test) is used to evaluate the effect of increase of the number of muscle synergies. Statistical significance level p is set to 0.05.

2.2 Musculoskeletal Model

This study focuses on sagittal movement of standing-up motion, and the human body is modelled by four segments of shank, thigh, pelvis, and HAT (head, arm and trunk). Figure 2a illustrates the developed skeletal model. Joint angles $\theta_{k(k=1,2,3,4)}$ are defined as the angle from the distal segment. Segment length, position of center of mass, mass, moment of inertia are defined as L_k, L_k^{G}, M_k, and I_k. Given these parameters, equation of motion is expressed by

$$\mathbf{I}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})\ddot{\boldsymbol{\Theta}} + \mathbf{h}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) + \mathbf{g}(\boldsymbol{\Theta}) = \mathbf{T}_{\mathrm{JNT}} + \boldsymbol{\Phi}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}), \tag{5}$$

$$\mathbf{T}_{\mathrm{JNT}} = \mathbf{T}_{\mathrm{MUS}} + \mathbf{T}_{\mathrm{POS}}, \tag{6}$$

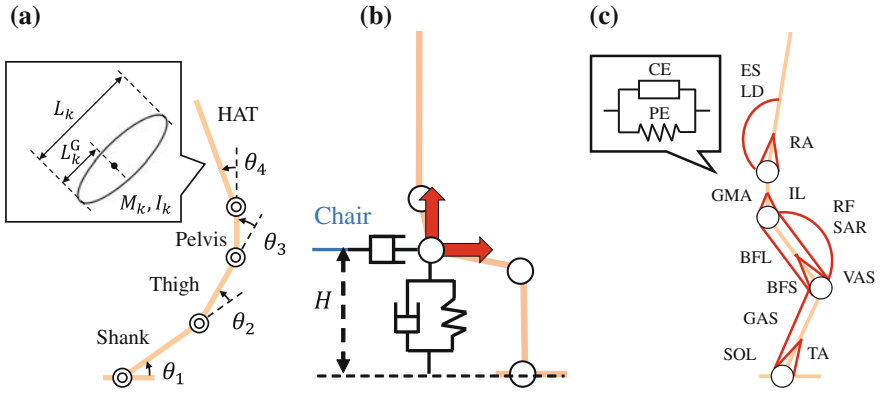


Fig. 2 Developed Musculoskeletal Model. **a** Shows four segment model of human body: shank, thigh, pelvis, and HAT. **b** Shows the floor model which applies horizontal and vertical forces on the hip joint with viscous and elastic elements. **c** Shows considered 13 muscles. Hill type model is used to represent muscles. **a** Skeletal model. **b** Floor model. **c** Muscle model

Table 1 Parameters for skeletal model below shows segment parameters of shank, thigh, pelvis, and HAT segments

	L_K (m)	M_K (m)	L_K^{G} (m)	I_K (m)
Shank	0.5	4.8	0.15	0.04
Thigh	0.4	9.6	0.16	0.13
Pelvis	0.1	17.6	0.01	0.05
HAT	0.7	48.0	0.14	3.55

where $\mathbf{I}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) \in \mathbb{R}^{4\times4}$, $\mathbf{h}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) \in \mathbb{R}^{4\times1}$, $\mathbf{g}(\boldsymbol{\Theta}) \in \mathbb{R}^{4\times1}$ indicate an inertia matrix, a non-linear term, and a gravitational force term. They are obtained from Lagrange equation. Vector $\boldsymbol{\Theta} \in \mathbb{R}^{4\times1}$ represents joint angles. Component of $\boldsymbol{\Theta}$ is each joint angle θ_k. Vector $\boldsymbol{\Phi}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) \in \mathbb{R}^{4\times1}$ indicates horizontal and vertical reaction force generated from elastic and viscous elements as shown in Fig. 2b. Reaction force is applied to the hip joint when the hip position is lower than the chair height H. Vector $\mathbf{T}_{\mathrm{JNT}} \in \mathbb{R}^{4\times1}$ indicates joint torque, and it consists of muscular torque ($\mathbf{T}_{\mathrm{MUS}} \in \mathbb{R}^{4\times1}$) and posture stabilization torque ($\mathbf{T}_{\mathrm{POS}} \in \mathbb{R}^{4\times1}$) as in Eq. (6). Vector $\mathbf{T}_{\mathrm{MUS}}$ is muscular joint torque calculated from the muscle model as explained below. Vector $\mathbf{T}_{\mathrm{POS}}$ is the joint torque to stabilize posture, and it is determined from PD control to follow the desired trajectories. Table 1 shows detailed parameters for body segments. Segment parameters are decided from measured data and anatomical data of a human body [7]. Also, coefficient of elastic elements of reaction force was 10,000 N/m for the vertical direction. Coefficients of viscous elements were set to be 300 and 400 Ns/m for horizontal and vertical directions. The chair height H was set to be 0.555 m.

Thirteen muscles in lower body and trunk which either flex or extend the body segments are considered including mono and bi articular muscles (Fig. 2c): tibialis anterior (TA), gastrocenomius (GAS), soleus (SOL), rectus femoris (RF), vastus lateralis (VAS), biceps femoris long head (BFL), biceps femoris short head (BFS), gluteus maximus (GMA), sartorius (SAR), rectus abdominis (RA), elector spine (ES), latissimus dorsi (LD), and iliopsoas (IL).

Vector $\mathbf{T}_{\mathrm{MUS}}$ is generated from individual muscles and it is calculated from moment arm of muscles and muscular tension by

$$\mathbf{T}_{\mathrm{MUS}} = \mathbf{A}\mathbf{F}(l_i, \dot{l}_i, m_i), \tag{7}$$

$$\mathbf{A} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{41} & \cdots & r_{4n} \end{pmatrix}. \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{4\times n}$ is a moment arm matrix that indicates the moment arm length of muscles to joints. Its component r_{ki} indicates the moment arm length of ith muscle on the joint k. Moment arm length r_{ki} is zero if the muscle i does not attach to the

joint k, and otherwise it is either positive or negative depending on either the muscle extends or flexes the ankle, knee, hip, and pelvis joints. In this study, moment arm is supposed to be constant regardless of body posture $\boldsymbol{\Theta}$.

Vector $\mathbf{F} \in \mathbb{R}^{n\times 1}$ indicates muscular tension. Muscle is represented by the Hill type model [8] (Fig. 2c). Using the model, muscular force of ith muscle F_i is generated from two components: contractile element (CE) and parallel elastic element (PE). F_i is given by

$$F_i(l_i, \dot{l}_i, m_i) = F_i^{\mathrm{CE}} + F_i^{\mathrm{PE}}, \tag{9}$$

$$F_i^{\mathrm{CE}} = F_i^{\max} f_{\mathrm{fl}}(\widetilde{l}_i) f_{\mathrm{fv}}(\widetilde{v}_i) m_i, \tag{10}$$

$$F_i^{\mathrm{PE}} = F_i^{\max} f_{\mathrm{pe}}(\widetilde{l}_i), \tag{11}$$

$$\widetilde{l}_i = (l_i^o + \sum_{k=1}^{4} (r_{ki}(\theta_k - \theta_o))/l_i^o, \tag{12}$$

$$\widetilde{v}_i = \frac{1}{10 l_i^o} \frac{dl_i}{dt}. \tag{13}$$

where F_i^{CE} is the force generated by CE that is calculated from maximum isometric force ($F_i^{\max}$), muscle activation (m_i), and muscle dynamics (f_{fl} and f_{fv}). When normalized muscle length $\widetilde{l}_i$ and normalized muscular velocity $\widetilde{v}_i$ are given, force-length relationship $f_{\mathrm{fl}}(\widetilde{l}_i)$ and force-velocity relationship $f_{\mathrm{fv}}(\widetilde{v}_i)$ are considered as muscle dynamics [9, 10]. On the other hand, PE generates force when the muscle length l_i is longer than its optimal length as in Eq. (11). The force of PE F_i^{PE} is calculated according to the normalized muscle length $\widetilde{l}_i$ [11]. The normalized muscle length $\widetilde{l}_i$ is decided from moment arm and changes of the current joint angle from the optimal joint angle θ_o (Eq. (12)) [12]. The optimal joint angles is equal to the posture when humans stand vertically. The normalized muscular velocity $\widetilde{v}_i$ is obtained from Eq. (13). The parameters for muscle models were determined from previous studies [12–14].

Vector $\mathbf{T}_{\mathrm{POS}}$ is generated from PD control to follow the target kinematics and is given by

$$\mathbf{T}_{\mathrm{Pos}}(t) = \begin{cases} 0 & \text{when } t < \lambda_{\mathrm{delay}} \\ \mathbf{K}_{\mathrm{P}}^{\mathrm{q}} \Delta\mathbf{q}(t-\lambda) + \mathbf{K}_{\mathrm{D}}^{\mathrm{q}} d\Delta\mathbf{q}(t-\lambda) + \mathbf{K}_{\mathrm{D}}^{\dot{\mathrm{q}}} d\Delta\dot{\mathbf{q}}(t-\lambda) & \text{when } t \geq \lambda_{\mathrm{delay}}, \end{cases} \tag{14}$$

$$\Delta\mathbf{q}(t) = \hat{\mathbf{q}}(t) - \mathbf{q}(t), \tag{15}$$

$$\Delta\dot{\mathbf{q}}(t) = \hat{\dot{\mathbf{q}}}(t) - \dot{\mathbf{q}}(t), \tag{16}$$

where $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ indicate the model's joint angle and angular velocity from the horizontal direction. On the other hand, $\hat{\mathbf{q}}(t)$ and $\hat{\dot{\mathbf{q}}}(t)$ are targeted kinematics. Additionally, nervous transmission delay λ is considered in this study to calculate $\mathbf{T}_{\mathrm{POS}}$. Gains for proportional and derivative control are manually determined as shown in Table 2. Nervous transmission delay λ was set to be 100 ms.

Table 2 Parameters for postural control

	K_P^q	K_D^q	$K_D^{\dot{q}}$
Foot	250	33,500	1,500
Knee	350	43,500	1,000
Hip	80	1,570	70
Lumbar	400	41,000	2,500

Below shows parameters of skeletal model and for PD controller of the foot, knee, hip and lumbar joints

2.3 Forward Dynamics Simulation

Forward dynamic simulation is conducted to generate movement with the developed musculoskeletal model. Before the simulation, body kinematics, reaction force, and muscle activation are averaged for all data in a measurement experiment explained in the next section. In order to generate the standing-up motion, firstly joint torque $\mathbf{T}_{\mathrm{JNT}}$ is calculated using inverse dynamics from the average body kinematics and reaction force. Then the joint torque $\mathbf{T}_{\mathrm{JNT}}$ is decomposed to the muscle activation $\mathbf{m}_i$ which can necessarily generate the joint torque for the standing-up motion. Since our musculoskeletal model has bi-articular muscles, the muscle activation cannot be determined uniquely from the given joint torque. Therefore in this study, the muscle activation $\mathbf{m}_i$ is decided through optimization methodology to minimize squared error between muscle activation $\mathbf{m}_i$ and mean of measured muscle activation $\hat{\mathbf{m}}_i$.

Muscle synergy $\mathbf{w}_j$ and time-varying weighting coefficient $\mathbf{c}_j$ are calculated from muscle activation $\mathbf{m}$ to generate the standing-up motion. In particular, time-varying weighting coefficient is expressed in a trapezoid wave. During the simulation $\mathbf{T}_{\mathrm{JNT}}$ consists of muscular torque $\mathbf{T}_{\mathrm{MUS}}$ and posture stabilization torque $\mathbf{T}_{\mathrm{POS}}$. Muscular torque is generated from $\mathbf{w}_j$ and $\mathbf{c}_j$ and from body posture $\boldsymbol{\Theta}$ and $\dot{\boldsymbol{\Theta}}$. Posture stabilization torque ($\mathbf{T}_{\mathrm{POS}}$) is decided to follow the average body kinematics. The average joint angles and angular velocities at the start of the data are used for initial posture of the simulation. For numerical calculation, fourth order Runge-Kutta method is used with time step of 1 ms.

2.4 Empirical Experiment with Humans

The measurement experiment was conducted to use in the inverse dynamics of the musculoskeletal model and to evaluate the results of forward dynamic simulation. One young healthy male participated in the experiment (age: 27 years, height: 1.76 m, weight: 77 kg). Muscle activation was measured in 1,000 Hz by DL-720 (S&ME Corp.). In the experiment, 17 body positions were measured according to Helen Hayes marker set, and they were obtained in 200 Hz using an optical motion capture system MAC3D (MotionAnalysis Corp.). Reaction force from participant's

hip was measured in 64 Hz using force sensors (Nitta Corp.). Consent was obtained before starting the experiment, and this study was conducted with approval by the Institute Review Board (IRB) of the University of Tokyo.

Muscle activation data was filtered with 10 Hz high pass and 200 Hz low pass second order butterworth filter. Also muscle activation data was rectified and normalized based on maximum voluntary contraction (MVC). Body position and reaction force data were filtered with low-pass second order butterworth filter with cut-off frequency 10 Hz and 20 Hz respectively. Joint angles $\theta_{1,2,3,4}$ were calculated using SIMM (MusculoGraphics Corp.).

Our experiment consisted of two trials. Each trial continued for 150 s and the participant was asked to repeat the sit-to-stand and stand-to-sit motion during the trial. We used measured data only during the sit-to-stand motion. The chair height was adjusted to the length of his shank segment. During the experiment, the participant was asked to cross their arms in front of their chest in order to avoid the use of their hands and arms. The ankle joint angle was set vertically to the ground at the start of the motion, and the participant was told not to move their feet during measurement. Moreover, the subject was asked to perform the standing-up motion in a comfortable speed.

2.5 Analysis of Standing-up Motion

In order to investigate the relationship between muscle synergies and body kinematics, four kinematic important phases are focused. It is known that there are four characteristic events (phases 1–4) in human standing-up motion [15]. Figure 3 illustrates movement during four phases in standing-up motion and they are described as follows.

1. Phase 1: Bending trunk forward to generate momentum.
2. Phase 2: Rising hip to move the center of mass forward.
3. Phase 3: Extending body to move the center of mass upward.
4. Phase 4: Stabilizing posture.

The start point of each phase is determined from measured body kinematics and reaction force data. The start of phase 1 is decided when humans bend their trunk forward. Therefore it is decided as the horizontal shoulder velocity exceeds the threshold p_1. Phase 2 begins when humans rise their hip, and the start time is obtained from the time when reaction force of hip is below the threshold p_2. Phase 3 starts when extension of body starts after the forward movement of the center of mass. The start time of phase 3 is obtained when the horizontal knee position reaches the most front point. At last, the start point of phase 4 is obtained when humans complete the standing-up motion. It is calculated from the time when vertical shoulder velocity is below the threshold p_4. Measured data 1.0 s before and 1.0 s after the time of hip rising is used for analysis.

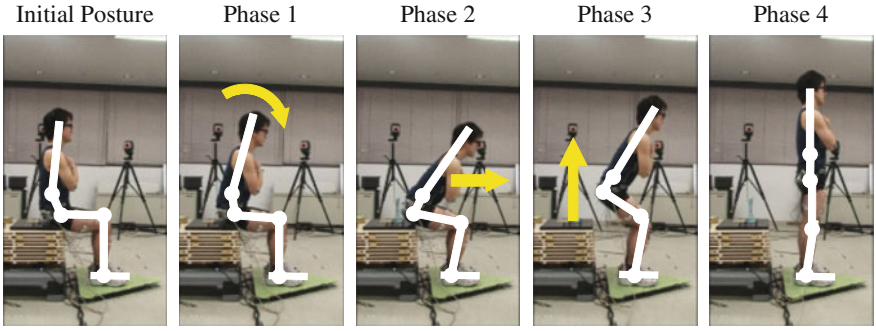


Fig. 3 Four Kinematic Phases in Standing-up Motion. In phase 1, humans start bending. In phase 2, humans move their center of mass forward. In phase 3, humans extend their body to move upward. In phase 4, humans decelerate the movement of the center of mass

3 Results

3.1 Results of Measured Standing-up Motion

From the measurement experiment, 28 trials of standing-up motion were obtained. Thresholds to decide the phase start were set to 0.2 m/s, 5 N, and 0.0 m/s respectively for p_1, p_2, and p_4. In all obtained trials, average and standard deviation of start times of four phases were 0.29 ± 0.07 s, 1.00 s, 1.42 ± 0.10 s, and 1.70 ± 0.25 s respectively for phases 1–4.

3.2 Results of Muscle Synergy from Measured Data

Figure 4 shows the average and standard deviation of the coefficient of determination R^2 for different numbers of muscle synergies. Muscle synergies $\mathbf{w}_j$ and time-varying weighting coefficient $\mathbf{c}_j$ were calculated from $\mathbf{m}$. Coefficients of determination exceed 95 % of the variance when the number of muscle synergies was four. Moreover, ANOVA revealed a statistical significance on the coefficient of determination according to the number of muscle synergies. Therefore a post-hoc test was applied to the neighbouring number of synergies to investigate whether additional synergies could increase the coefficient of determination. Results showed a statistical significance between the number of muscle synergies one and two, two and three, and three and four. Therefore, the number of muscle synergies was set to be four.

Figure 5a, c, e, g illustrate muscle activation level included in extracted muscle synergies 1–4. Figure 5b, d, f, h illustrate time-varying weighting coefficients corresponded to four different synergies. In the figures, vertical lines respectively show start time of phases 1–4.

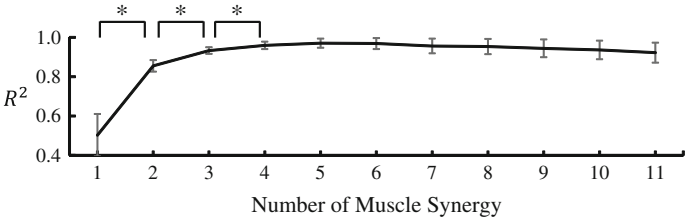


Fig. 4 Results of coefficient of determination. Results of statistical analysis shows that there was statistical significance between the number of muscle synergies one and two, two and three, and three and four. Moreover, four muscle synergies could explain more than 95 % of muscle activation

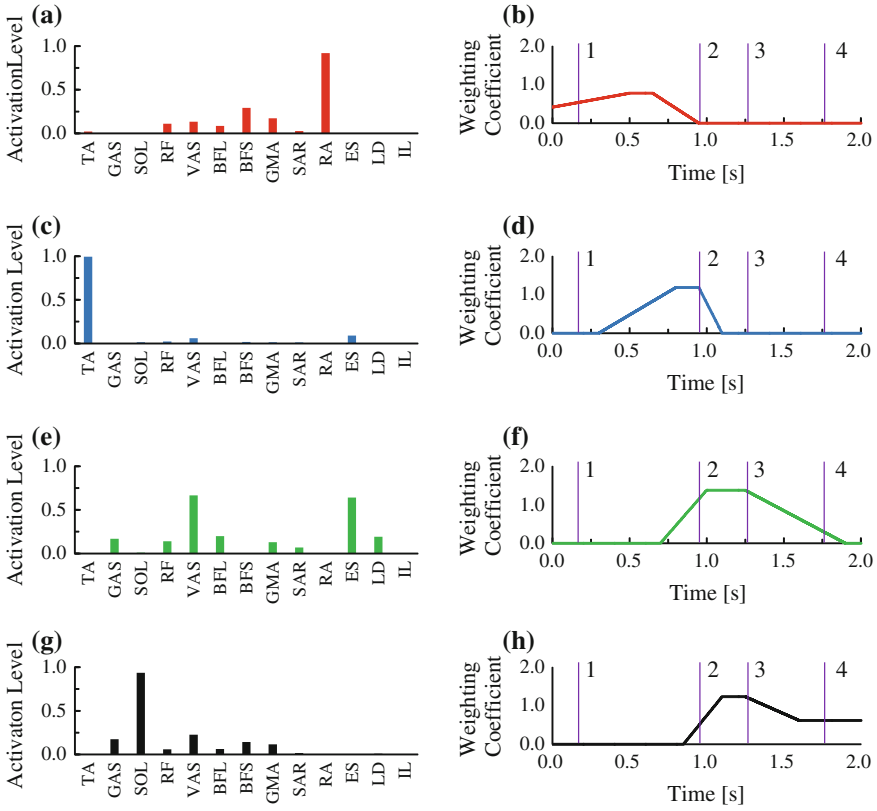


Fig. 5 Muscle synergy of forward dynamics simulation. **a**, **c**, **e**, and **g** show relative excitation muscle activation included in each synergy. **b**, **d**, **f**, and **h** show time-varying weighting coefficients of muscle synergies. The vertical lines show start time of four different phases. **a** Muscle synergy 1. **b** Muscle synergy 1. **c** Muscle synergy 2. **d** Muscle synergy 2. **e** Muscle synergy 3. **f** Muscle synergy 3. **g** Muscle synergy 4. **h** Muscle synergy 4

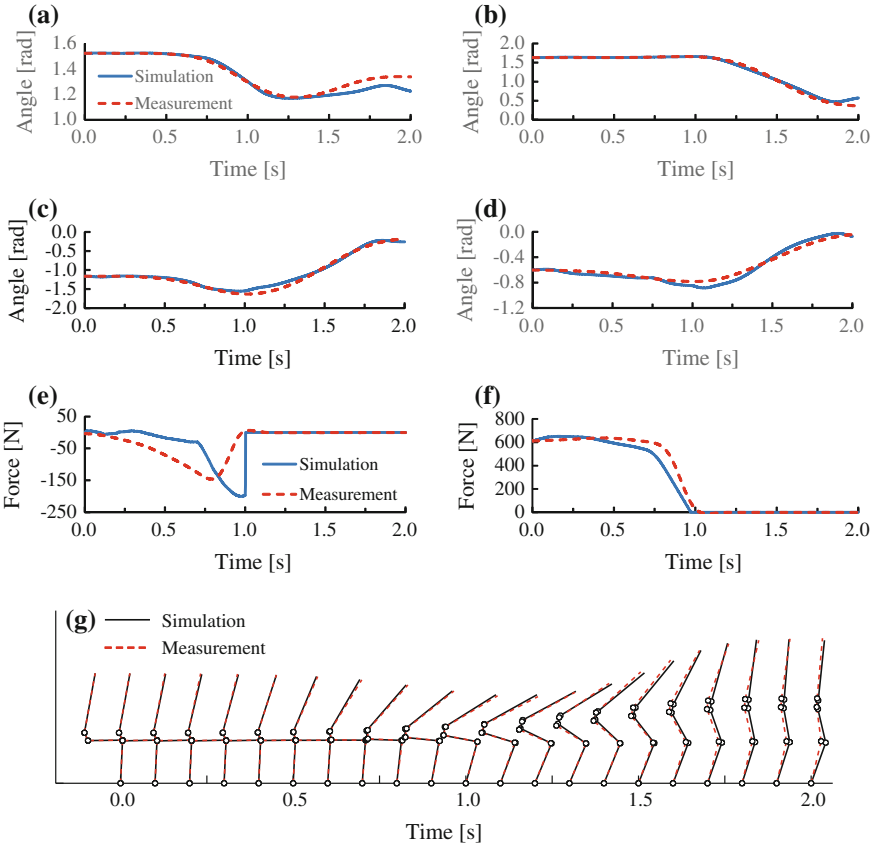


Fig. 6 Forward dynamics simulation results. **a**–**d** show comparison of joint angles (θ_1–θ_4) between simulated ones (*solid lines*) and measured ones (*dashed lines*). **e**–**f** show comparison of horizontal and vertical reaction force between simulated ones (*solid lines*) and measured ones (*dashed lines*). **g** shows stick pictures of simulated standing-up motion. **a** Ankle joint angle θ_1. **b** Knee joint angle θ_2. **c** Hip joint angle θ_3. **d** Lumbar joint angle θ_4. **e** Horizontal reaction force. **f** Vertical reaction force. **g** Stick pictures of simulated standing-up motion

3.3 Results of Forward Dynamics Simulation

Figure 6a–d show comparison between simulated joint angles and measured angles of the ankle, knee, hip, and lumbar joints respectively. Figure 6e–f shows comparison between simulated and measured reaction force in horizontal and vertical directions. Figure 6g shows the simulated movement of the musculoskeletal model performing standing-up motion.

4 Discussion

Four muscle synergies were obtained from the measured muscle activities during standing-up motion. Muscle excitation level of muscle synergies and time-varying weighting coefficients showed that each synergy had different contribution to human standing-up motion and it corresponded to characteristic four phases of standing-up motion reported previously [15].

Muscle synergy 1 involved muscle activation of RA and it was mainly activated during phase 1. This implied that humans activated RA to flex their trunk for forward bending (generating momentum). Similarly, muscle synergy 2 was activated the most at the start time of phase 2. Muscle synergy 2 was mainly contributed by TA. It corresponded to the movement during phase 2 to move the center of mass forward by dorsiflexion of the ankle by TA. On the other hand, VAS and ES were activated in muscle synergy 3 to extend their knee and trunk. This muscle synergy was mainly activated during phase 3 to extend whole body and lift up the center of mass upward. At last, activation of muscle synergy 4 was mainly seen in phase 4. In the synergy 4, SOL was activated to extend ankle joint to decelerate the horizontal movement of center of mass.

Different from our previous study [4], four muscle synergies have been extracted from human standing-up motion instead of three muscle synergies. Since our previous study only considered lower body muscles, it could not fully express the movement of trunk. However, this study has included additional muscles in trunk, and therefore the muscle synergy 1 was newly obtained. Although most of the studies to analyze standing-up motion considered only three joint angles of ankle, knee and hip, this study developed musculoskeletal model of four rigid body segments according to the anatomical contribution of each muscle on body joints. Our forward dynamics simulation results showed that four muscle synergies could achieve a dynamically plausible standing-up motion.

5 Conclusions and Future Works

Four essential muscle synergies were extracted from measured muscle activation of lower body and trunk during the human standing-up motion. Muscles activation level involved in each muscle synergy and its time-varying weighting coefficients correspond to characteristic body movement of standing-up motion. Moreover the musculoskeletal model was developed considering dynamics and anatomical characteristics of human body. Our forward dynamics simulation showed that four muscle synergies could successfully achieve the human standing-up motion instead of controlling individual muscles.

One of our future direction will be study of how the obtained muscle synergies are robust for environmental changes. For example, it is necessary to investigate whether four muscle synergies can realize the motion from different chair seat heights

or different feet positions. Another interesting direction is to analyze structure of muscle synergies. In the current study, four muscle synergies are obtained, but it is unclear whether each synergy works independently or one synergy is dependent on another synergy. Using the developed musculoskeletal model, relationship between each muscle synergy will be studied.

Acknowledgments This work was in part supported by JSPS KAKENHI Grant Number 26120005 and 26120006, the MEXT KAKENHI, Gtant-in-Aid for Scientific Research (B) 24300198, JST RISTEX Service Science, Solutions and Foundation Integrated Research Program, and Grant-in-Aid for JSPS Fellows 24·8702.

References

1. Bernstein, N.: The Co-ordination and Regulation of Movement. Pergamon, Oxford (1967)
2. Ivanenko, Y.P., Poppele, R.E., Lacquaniti, F.: Five basic muscle activation patterns account for muscle activity during human locomotion. J. Physiol. **556**, 267–282 (2004)
3. d'Avella, A., Bizzi, E.: Shared and specific muscle synergies in natural motor behavior. Proc. Natl. Acad. Sci. **102**, 3076–3081 (2005)
4. An Q, Ishikawa Y, Funato T, Aoi S, Oka H, Yamakawa H, Yamashita A, Asama H: Generation of human standing-up motion with muscle synergies using forward dynamic simulation. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA2014), pp. 730–735. Hong Kong (China), June 2014
5. Lee, D.D., Seun, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)
6. Ting, L.H., Macpherson, J.M.: A limited set of muscle synergies for force control during a postural task. J. Neurophysiol. **93**, 609–613 (2005)
7. Clauser, C.E., McConville, J.T., Young, J.W.: Weight, Volume, and Center of Mass of Segments of Human Body, pp. 69–70. Wright-Patterson Air Force Basem Ohion, AMRL Technical Report (1969)
8. Zajac, F.E.: Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. Crit. Rev. Biomed. Eng. **17**, 359–411 (1989)
9. Hatze, H.: Myocybernetic control models of skeletal muscles. Biol. Cybern. **25**, 103–119 (1977)
10. Ogihara, N., Yamazaki, N.: Generation of human bipedal locomotion by a bio-mimetic neuro-musculo-skeletal model. Biol. Cybern. **84**, 1–11 (2001)
11. Kuo, P., Deshpahde A.D.: Contribution of passive properties of muscle-tendon units to the metacarpophalangeal joint torque of the index finger. In: Proceedings of the 2010 IEEE RAS&EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob2010), pp. 288–294 (2010)
12. Riener, R., Fuhr, T.: Patient-driven control of FES-supported standing up, a simulation study. IEEE Trans. Rehabil. Eng. **6**, 113–124 (1998)
13. Arnold, E.M., Ward, S.R., Lieber, R.L., Delp, S.L.: A model of the lower limb for analysis of human movement. Ann. Biomed. Eng. **38**, 269–279 (2010)
14. Jorgensen, M.J., Marras, W.S., Granata, K.P., Wian, J.W.: MRI-derived moment-arms of the female and male spine loading muscles. Clin. Biomech. **16**, 182–193 (2001)
15. Schenkman, M., Berger, R.A., Patrick, O.R., Mann, R.W., Hodge, W.A.: Whole-body movements during rising to standing from sitting. Phys. Ther. **70**, 638–651 (1990)