

Studies in Computational Intelligence 604

Anis Koubâa

J. Ramiro Martínez-de Dios *Editors*

Cooperative Robots and Sensor Networks 2015

 Springer

Studies in Computational Intelligence

Volume 604

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series “Studies in Computational Intelligence” (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at <http://www.springer.com/series/7092>

Anis Koubâa · J. Ramiro Martínez-de Dios
Editors

Cooperative Robots and Sensor Networks 2015

 Springer

Editors

Anis Koubâa
College of Computer and Information
Sciences (CCIS)
Prince Sultan University
Riyadh
Saudi Arabia

J. Ramiro Martínez-de Dios
Departamento de Ingeniería de Sistemas y
Automática Escuela Superior de
Ingenieros
University of Seville
Sevilla
Spain

and

CISTER/INESC-TEC, ISEP
Polytechnic Institute of Porto
Porto
Portugal

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-319-18298-8 ISBN 978-3-319-18299-5 (eBook)
DOI 10.1007/978-3-319-18299-5

Library of Congress Control Number: 2015938090

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Contents

Part I Multi-Robots Systems

COROS: A Multi-Agent Software Architecture for Cooperative and Autonomous Service Robots	3
Anis Koubâa, Mohamed-Foued Sriti, Hachemi Bennaceur, Adel Ammar, Yasir Javed, Maram Alajlan, Nada Al-Elaiwi, Mohamed Tounsi and Elhadi Shakshuki	
Multi-robot Task Allocation: A Review of the State-of-the-Art.	31
Alaa Khamis, Ahmed Hussein and Ahmed Elmogy	
Efficient Trajectory Planning for WSN Data Collection with Multiple UAVs.	53
D. Alejo, J.A. Cobano, G. Heredia, J. Ramiro Martínez-de Dios and A. Ollero	
Multi-robot Surveillance Through a Distributed Sensor Network.	77
Andrea Pennisi, Fabio Previtali, Cristiano Gennari, Domenico D. Bloisi, Luca Iocchi, Francesco Ficarola, Andrea Vitaletti and Daniele Nardi	

Part II Data Fusion, Localization and Mapping

Exploiting Multi-hop Inter-beacon Measurements in RO-SLAM	101
A. Torres-González, J. Ramiro Martínez-de Dios and A. Ollero	
A Distributed and Multithreaded SLAM Architecture for Robotic Clusters and Wireless Sensor Networks	121
David Portugal, Bruno D. Gouveia and Lino Marques	

A Particle Filter-Based Method for Ground-Based WSN Localization Using an Aerial Robot	143
Francisco Cuesta, Miguel Cordero, Luis Díaz, Antidio Viguria and Aníbal Ollero	
Fundamental Limits of Self-localization for Cooperative Robotic Platforms Using Signals of Opportunity	159
Mei Leng and Wee Peng Tay	
Part III Security and Dependability	
Security in Mobile Wireless Sensor Networks: Attacks and Defenses	185
Amrita Ghosal and Subir Halder	
Dependable Communication for Mobile Robots in Industrial Wireless Mesh Networks	207
Timo Lindhorst and Edgar Nett	
Part IV Mobility	
Automatic Merging of Vehicles: Design, Algorithms, Performance	231
Gurulingesh Raravi, Vipul Shingde and Krithi Ramamritham	
A Survey on Data Collection in Mobile Wireless Sensor Networks (MWSNs)	257
Ali Sayyed and Leandro Buss Becker	

About the Book

In the recent years, different technological fields have emerged in the context of ubiquitous systems. Technologies such as personal mobile computing, camera networks, wearable computing, Radio-Frequency IDentification (RFID) devices, or Wireless Sensor Networks (WSN) have been integrated with a wide range of existing technologies such as robotics. The cooperation between mobile robots and sensor networks offer unprecedented possibilities in a wide range of problems and applications. Interoperability between them, from intrinsically different technological fields and broadly understood heterogeneous, pose key problems that are still far from being solved.

The book you have in your hands compiles some of the latest research in cooperation between robots and sensor networks. Structured into 12 chapters, this book addresses fundamental, theoretical, implementation, and experimentation issues. Chapters are organized into four parts, namely Multi-Robot Systems, Data Fusion and Localization, Security and Dependability, and Finally Mobility.

We hope that this book may spark innovative ideas and improvements in your research and applications.

Part I
Multi-Robots Systems

COROS: A Multi-Agent Software Architecture for Cooperative and Autonomous Service Robots

Anis Koubâa, Mohamed-Foued Sriti, Hachemi Bennaceur,
Adel Ammar, Yasir Javed, Maram Alajlan, Nada Al-Elaiwi,
Mohamed Tounsi and Elhadi Shakshuki

Abstract Building distributed applications for cooperative service robots systems is a very challenging task from software engineering perspective. Indeed, apart from the complexity of designing software components for the control of a single autonomous robot, cooperative multi-robot systems require additional care in the design of software components to ensure communication and coordination between the robotic

A. Koubâa (✉) · Y. Javed · M. Tounsi
Prince Sultan University, Riyadh, Saudi Arabia
e-mail: akoubaa@coins-lab.org

Y. Javed
e-mail: yasir.javed@coins-lab.org

M. Tounsi
e-mail: mtounsi@cis.psu.edu.sa

M.-F. Sriti · H. Bennaceur · A. Ammar · M. Alajlan
College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic
University (IMSIU), Riyadh, Kingdom of Saudi Arabia
e-mail: mfsriti@ccis.imamu.edu.sa

H. Bennaceur
e-mail: hachemi@ccis.imamu.edu.sa

A. Ammar
e-mail: adel.ammar@ccis.imamu.edu.sa

M. Alajlan
e-mail: maram.ajlan@coins-lab.org

A. Koubâa · Y. Javed · M. Alajlan · N. Al-Elaiwi
COINS Research Group, Riyadh, Saudi Arabia

A. Koubâa
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal

E. Shakshuki
Acadia University, Wolfville, Canada
e-mail: elhadi.shakshuki@acadiau.ca

N. Al-Elaiwi
King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia
e-mail: nalelaiwi@kacst.edu.sa

© Springer International Publishing Switzerland 2015

A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604, DOI 10.1007/978-3-319-18299-5_1

agents. This chapter proposes COROS, a new multi-agent software architecture for cooperative and autonomous service robots with the objective to make easier the design and development of multi-robot applications. We present a high-level conceptual architecture for multi-agent robotics systems that represents a generic framework for cooperative multi-robot applications. Furthermore, we present an instantiation of this generic architecture with an implementation software architecture on top of the Robot Operating System (ROS) middleware. The proposed concrete software architecture follows a component-based approach to ensure modularity, software reuse, extensibility and scalability of the multi-robot operational software. In addition, one major added value of our architecture is that it provides a tangible solution to supporting multi-robot software development for the ROS middleware, as ROS was originally designed for single-robot applications. We also demonstrate a sample of real-world case studies of cooperative and autonomous service robots applications in an office-like environment, including discovery and courier delivery applications.

Keywords Autonomous service robots · Cooperative robots · Robotic software engineering · Multi-agent systems · Robot operating system (ROS)

1 Introduction

The design of efficient software architecture for service robots (e.g. home automation, indoor surveillance, elderly people care, etc.) is an increasingly important issue in robotics research considering the expansion of their market. Indeed, the demand for these service robots is increasing as the International Federation of Robotics reported in its statistics that 3 million service robots for personal and domestic use were sold in 2012, which represents 20% more than in 2011, increasing sales up to US\$1.2 billion [1]. The success of deployment of service robots at public scale is tightly coupled with the efficiency of software design for service robots applications. A first major challenge with respect to robotic software engineering is the heterogeneity of robotic hardware platform in addition to the absence of standards. One common solution to this problem is the use of robotic middlewares that provide abstraction layers to robotic sensors and actuators. There has been several initiatives for robots middlewares including the Player/Stage project [2], Middleware for Robotic Applications (MIRA) cross-platform framework [3], the Mobile Robot Programming Toolkit (MRPT) [4], and the widely-used Robot Operating System (ROS) [5]. While these middlewares are effective in helping applications developers avoiding the programming complexity of low-level hardware components, they do not provide sufficient abstractions to design complex applications for service robots. For that purpose, several high-level robotic software architectures [6–9] were proposed in the literature to provide a conceptual view of software components and their interactions needed for specifying robotic services. Most of the aforementioned works and middlewares

addressed the architectural design for single robot applications, which might present limitations when applied to multi-robot systems. Indeed, multi-robot systems are typically more challenging in terms of software engineering since they exhibit additional requirements as compared to single robot systems, including communication and coordination to accomplish their missions. Some previous works (e.g. [10]) have also proposed system architectures for multi-robot applications.

In this book chapter, we consider the problem of designing software architecture for cooperative multi-robot applications. Indeed, in the context of iroboapp project [11], we have been working on designing intelligent applications for single and multiple robots in two main research directions, namely, (1) global path planning and (2) the multi-robot task allocation (MRTA) problem. In this project, we conducted a detailed study of the performance of a set of meta-heuristics for single and multi-robots path planning, and multi-robot task allocation. This study was completed with the design of several techniques for these problems. Many simulations of these techniques have been carried out to prove their efficiency. One of our objectives of the project is to validate our findings through real-world implementations and experimentation on robots, which led us to design a software architecture for multi-robots applications. We opted for the use of ROS as programming middleware to have an abstraction layer on top of robotic hardware, sensors and actuators. However, we found out that ROS was designed for single robots applications and several challenges were faced to efficiently implement multi-robot applications on top of ROS. This conducted us to re-think about efficient implementation strategies to support multi-robot in the ROS middleware. As an example, ROS is heavily based on concept of topics which represent a particular stream of data that might be associated to a sensor device (e.g. laser data, camera images, ...), actuator (e.g. motor status), or any other program-logic (grid map, user-defined data). Topics basically describe the internal status of the robot, and in the case of cooperative robot applications, there is a need to exchange robots' mutual status. However, ROS does not natively support this kind of interaction. In this work, we provide a solution to this problem in two steps. We first analyse the high-level conceptual requirements of multi-agent robotics systems. Then, we present COROS which is a generic software architecture for ROS-enabled robot that promotes the effectiveness of building new distributed robotic applications.

The summary of contributions are four-folded. First, we present and analyze, in Sect. 3, the state-of-the art of software engineering in robotics and the different approaches adopted to build complex software for single and multi-robots applications. Second, in Sect. 4, we propose COROS, a multi-agent software architecture, at both conceptual level and implementation level, that facilitates the design and development of multi-robot applications and promotes software reuse, modularity and extensibility. Third, we demonstrate how COROS can be integrated with ROS middleware. Finally, in Sect. 5 we present two real-world experimental scenarios to demonstrate the effectiveness of COROS architecture in development of real distributed multi-robot applications.

2 Background

2.1 Robotic Operating System (ROS)

In this section, we present a general overview of the basic concepts of ROS framework [5] to provide the required background needed to understand the software architecture proposed in this chapter. This overview is not intended to be comprehensive but just an introduction of the important concepts, and interested readers may refer to [12] for details.

ROS (Robot Operating System) has been developed, by Willow Garage [13] and Stanford University as a part of STAIR [14] project, as a free and open-source robotic middleware for the large-scale development of complex robotic systems.

ROS acts as a meta-operating system for robots as it provides hardware abstraction, low-level device control, inter-processes message-passing and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. The main advantage of ROS is that it allows manipulating sensor data of the robot as a labeled abstract data stream, called topic, without having to deal with hardware drivers. This makes the programming of robots much easier for software developers as they do not have to deal with hardware drivers and interfaces. It is useful to mention that ROS is not a real-time framework, though it is possible to integrate it with real-time code.

ROS relies on the concept of computational graph, which represents the network of ROS processes (potentially distributed across machines). An example of a simplified computation graph is illustrated in Fig. 1.

A process in ROS is called a *node*, which is responsible for performing computations and processing data collected from sensors. As illustrated in Fig. 1, a ROS system is typically composed of several nodes (i.e. processes), where each node

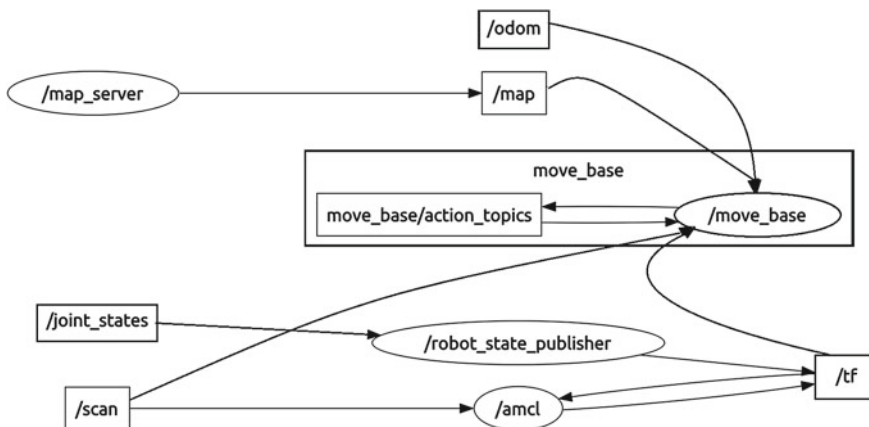


Fig. 1 Example of a ROS computation graph

processes a certain data. For example, *move_base* is a node that controls the robot navigation, *amcl* is another node responsible for the localization of the robot, and *map_server* is a node that provides the map of the environment to other processes of the system. Nodes are able to communicate through message-passing, where a *message* is a data structure with different typed-fields. This communication between nodes is only possible thanks to a central node, referred to as *ROS Master*, which acts as a name server providing name registration and lookup for all components of a ROS computation graph (e.g. nodes), and store relevant data about the running system in a central repository called *Parameter Server*.

ROS supports two main communication models between nodes:

- The *publish/subscribe* model: in this model nodes exchange *topics*, which represents a particular flow on data. One node or several nodes may act as a publisher(s) of a particular topic, and several nodes may subscribe to that topic, through the *ROS Master*. Subscriber and publisher nodes do not need to know about the existence between other because the interaction is based on the topic name and made through the *ROS Master*. For example, in Fig. 1, the *map_server* is the publisher of the topic */map*, which is consumed by the subscriber node *move_base*, which uses the map for navigation purposes. */scan* represents the flow of data received from the laser range finder, also used by *move_base* node to avoid obstacles. */odom* represents the control information used by *move_base* to control robot motion.
- The *request/reply* model: in this model, one node acts as a server that offers the service under a certain name, and receives and processes requests from other nodes acting as clients. Services are defined by a pair of message structures: one message for the request, and one message for the reply. Services are not represented in the ROS computation graph.

ROS and Multi-Robots: ROS was originally designed for single robot systems, but allows several robot machines or workstation machines to communicate through ROS topics in condition that one of the machines should run the *roscore* node, that is the node that identifies the *ROS Master*. This approach cannot be considered as distributed, although it allows to run several robots, because it relies on a central machine to run the *roscore*, which is prone to the typical single point of failure problem and is not scalable. There are some contributed ROS packages such as *foreign – relay* that relays the topic over multiple robots running different ROS Masters. Also a ROS package called *wifi_comm* based on *foreign – relay* has been developed allowing peer-to-peer communication between multiple robots. Another effort is the *ros – rt – wmp* ROS contributed package which aims at replicating whatever ROS topic or service in another computer wirelessly connected with the source without the need of sharing the same *ROS Master*. This approach is more scalable than other approaches but heavily based on a specific routing protocol called RT-WMP [15]. However, all these contributed works are still in their early phases and do not provide comprehensive solutions to develop complex multi-robot systems.

3 Related Works

Till today, there are no standard general purpose guidelines that can be used in the design and development of robotics software, which made development of service robotics system very complicated, inflexible, and increased integration complexity. In fact, robot software developers often experience a sense of frustration when they have to develop from scratch a new application, which is almost the same as several other releases for different projects, because they have not been able to capture and exploit the commonalities. Moreover, current robot programming languages are reaching their limits. They are not flexible and powerful enough to master the challenges imposed by the intended future applications, such as cooperative service robots. Consequently, there is a pressing need to engineer the software development process in order to design reusable robotic software systems and to implement flexible, modular, interoperable code. By using a composition of reusable building blocks, software infrastructures, and unified design techniques, it is possible to reduce the cost and time-to-market of robotic applications while preserving their efficiency, robustness, safety, and reliability [16]. Unfortunately, the adoption of such a software engineering approach by robotics researchers has been slow, impeded by the tradition of individual research groups crafting independent and incompatible solutions to common problems. Moreover, standardization is not yet fully reached, in particular for applications that go beyond industrial robotics, making difficult to realize an effective interoperability of solutions developed for different problems [17]. The complexity of service robots due to high interaction with humans, the integration of numerous sensors and the complexity of components require an easy, efficient and flexible integration and control, that is a better robot system architecture. Besides, the interaction between robot components require a good management at earlier stages for better synchronization. In the literature, there has been several approaches and attempts to address this gap. In what follows, we present a sample of these research efforts.

Component-Based Approaches: A variety of approaches for robotic software development have been proposed, due to the wide range of domains where robots are used, the many forms and functions that a robot can have and perform, and the diversity of robotics researchers [17]. In this scope, references [18, 19] proposed component-based architecture models for robotics middlewares designed for intelligent mobile robots. The advantage of component-based software engineering (CBSE) is that it allows software systems to be created and maintained at lower costs and with increased stability and extensibility through reuse of approved components in flexible software architectures [20], while such important features tend to be neglected in the race toward computational performance. The authors in [18] argued that robotics is particularly well-suited for and in need of component-based approaches. For that purpose, they introduced Orca, an open-source component-based software engineering framework for mobile robotics associated with a repository of free, reusable components for building mobile robotic systems. Several indoor and outdoor projects underway implement robotic systems using Orca components. Nevertheless, Orca does not prescribe a particular architecture. The architecture is rather defined by the

set of components that are chosen and the manner in which they are composed. As pointed out by [21], the CBSE approach has to be complemented with a careful analysis of the application domain, which should lead to the definition of stable data structures and interfaces to effectively achieve component reusability.

Iborra et al. [22] presented a summary of different software practices in the robotic domains, including domain engineering, reference architectures, and component-oriented development, then they focused on model-driven engineering design. They presented a successful implementation named Architectural Framework for Control Units (ACRoSeT) within a European project (EFTCoR) for climbing vehicles, using component-based design paradigm [18, 23, 24]. To overcome limitations of the component-oriented development, the authors implemented a model-driven architecture which proposes the use of models as the principal artifact for software development; a model being a simplified representation of reality that shows only the aspects that are of interest.

Robotic Software Development Frameworks: Calisi et al. [17] identified four key features that a framework for robotic software development must address: concurrency model, information sharing model, support tools, and interoperability. These features have a significant impact on the software development cycle. The observation of these key features led the authors to design OpenRDK, a framework for robotic software components with a multi-threaded multi-processes structure and a blackboard-type inter-module communication and data sharing. This framework supports data exchange among the modules as well as the ability to inspect data and processes and thus to build tools that suitably support debugging. It can be exploited to design the software for a wide class of robotic systems.

In [25], the authors proposed a new design patterns architecture to synchronize the interaction of different robot's components, named Task-State-Pattern. The authors were able to provide an abstract coordination interface and architecture. Such work can ease the job of designers and programmers of the robot systems. In [26], a new object oriented API for robot controls has been suggested which uses advanced concepts like sensor-based motions and multi-robot synchronization. Such an API was provided to realize some advanced motion control concepts, and to overcome some shortages in existing systems, such as in KRL, where multiple robots synchronization is difficult to achieve. The authors suggested a layered vendor-independent software architecture for industrial robot application development; an object-oriented framework which improves a series of shortages of current robotics languages, such as multi-robot cooperation and sensor-guided tasks.

In the SoftRobot research project [26], a consortium of academic and industrial partners analyzed the requirements of current and future applications of industrial robots. Based on this analysis, they developed a software architecture that enables object-oriented software development for industrial robot systems using general-purpose programming languages. This architecture allows specifying real-time critical operations of robots and tools, including advanced concepts like sensor-based motions and multi-robot synchronization.

Westhoff and Zhang [6] proposed a software architecture framework of multi-modal service robots based upon the Roblet-Technology which is evaluated on the service robot TASER of the TAMS Institute at the University of Hamburg. The objective of the authors was to enable the programmer to easily develop advanced applications. The framework was designed for networked applications by providing a layer that encapsulates network programming. Also, it supports component-oriented approach to easily integrate the existing solutions. Developing an application based on this software framework can be easily transferred to other robotic systems, and it enables the building of high-level applications easily. As the framework hides the network programming, it allows to upgrade the programming and testing of applications in service robotics.

A*STAR Social Robotics Laboratory (ASORO) developed a software architecture framework for service robots (SAFSR) [27] that can be configured and programmed using the Extensible Markup Language (XML) scripting language. Their framework consists of several key independent software modules, which are grouped into four layers: cognitive layer, execution and control layer, modality layer and device layer. The objective of [27] was to make SAFSR architecture flexible, extensible, and maintainable. SAFSR has been successfully applied to the development of three different service robots. After exhibiting the three service robots in various events and including them in several technical challenges, the authors concluded that these robots successfully demonstrated the features of the SAFSR; reducing time and complexity required to design and implement intelligent service robotics in different service domains.

Finally, Luzzana [28] proposed an integration between SCA (Service Component Architecture) and ROS that allows the developers to fully exploit the benefits of both approaches while mitigating their deficiencies. A bridge software component connects a ROS-based subsystem with a SCA-based subsystem in such a way that this integration is transparent to each other. The effectiveness of the proposed approach was demonstrated by applying it to a new mobile robot with a distributed computation architecture that enables the developers to experiment several different design approaches. In our work, we also aim at providing an abstraction layer to network programming to make easier the development of distributed robotics applications. The added value of the present work is that we propose a multi-agent component-based approach to build abstraction layers on top of the ROS middleware.

Software Architecture for Multi-Robots Systems DeLoach et al. [29] observed that while many recent agent-oriented architectures have been developed, there have been few attempts at applying high-level approaches to cooperative robotics systems design. Whereas using multi-agent approaches for cooperative robotics may provide some of the missing elements evidenced in many cooperative robotic applications, such as generality, adaptive organization, and fault tolerance. Hence, DeLoach et al. [29] applied the Multiagent Systems Engineering (MaSE) methodology to design high-level cooperative behaviors between autonomous and heterogeneous robots for search and rescue applications. MaSE [30] is a general purpose seven-step process

associated with a detailed sequence of inter-related graphically based models, which guides a system developer from an initial system specification through implementation of heterogeneous multiagent systems. It provides a top-down approach to building cooperative robotic systems instead of the behavior-based bottom up approach employed in traditional robotic implementations [31]. Besides, it was designed to be independent of any particular multiagent system architecture, programming language, or communication framework, which makes it fitting to implement cooperative robot applications. Nevertheless, contrary to standard robotic architectures, DeLoach et al. [29] focused on designing only high-level cooperative behaviors because they assumed that the low-level behaviors common to mobile robots already exist in libraries.

More specifically, Matson and DeLoach [32] proposed an organization-based multi-agent system model (OMAS) to overcome the problem of losing sensor capabilities in robots in dangerous environments. The goal was to build fault tolerant system and architectures that deal with detecting and handling sensor failure and faults, and calibration of sensors to adapt to unknown environmental conditions. The proposed model tolerates faults by managing the available hardware sensors as a group, focusing on managing their entire set of capabilities instead of simple brute force approach to sensor switching in cases of failure. The results show that the OMAS would successfully reorganize when a valid adaptation was possible. Although Matson and DeLoach [32] deals with intra-robotic capability, the organization model can also be applied to multiple robots working as a team where the sensor capabilities of one robot can fail over to another robot in a complete or partial manner.

Focusing on the multi-agent perspective, Silva et al. [33] presented a Gaia-based generic model for a multi-agent system based on mobile robotic platforms along with two distinct models derived from it, one for open space environment and the other one for indoor environment. Gaia [34, 35] is a software engineering methodology that excludes requirements elicitation and implementation focusing on analysis and design of the system, for designing multi-robot applications. This methodology is both general, in that it is applicable to a wide range of multi-agent systems, and comprehensive, in that it deals with both the macro-level and the micro-level aspects of systems. The objective of [33] was to demonstrate how robots applications designers could model their systems faster and simpler by adapting the GAIA methodology. The authors concluded that GAIA methodology provides a high level of abstraction when compared to similar methodologies. Nevertheless, as pointed out by DeLoach et al. [29], GAIA methodology falls short when defining the interactions between agents. For this reason, we did not consider GAIA methodology in our work, but we opted for a component-based generic software architecture for multi-robot application and we demonstrate how to integrate it with ROS middleware.

4 Software Architecture for Cooperative Robots

4.1 Cooperative Multi-Robot Software Requirements

The overall objective in the development of the software architecture is to facilitate the programming of distributed applications for cooperative and autonomous service robots.

As an illustrative scenario, consider a team of multiple autonomous robots deployed in an indoor environment (e.g. office, home) that assist humans in daily activities. Consider also that these robots should coordinate among each other for any new mission to be executed. For example, when a user sends a command to the robots' team to deliver a courier from one office to another, the robots should coordinate so that only the robot with the lowest cost will execute the mission. Although the scenario seems to be simple there are a lot of challenges to be addressed from software engineering perspective.

These challenges can be enumerated in the following functional requirements:

- **Communication:** this is an essential requirement for distributed robotic applications as robots need to communicate to be able to coordinate and exchange information among each other. The main problem in communication is the limited wireless coverage of the robots, and this prevents some robots to be reachable from others. As such, each robot will rely on the partial information it will be getting from the neighbor robots to take decision on mission execution. For example, with limited coverage, two non-neighbor robots may decide to execute the same mission because of lack of communication.
- **Problem Solving:** for any cooperative robots application, a consensus must be achieved among robots for any mission to be executed. As such, each mission represents a new problem that must be solved in a distributed manner in all the robots. In the above example, the problem was to find and select the robot with the lowest cost to execute the task. This requires a specific agent in each robot responsible for finding an effective solution, in a distributed manner, for any new mission shared among the robots.
- **Knowledge base:** In distributed applications, each robot should have its own knowledge base continuously gathering information about the current environment, other robots, and accomplished and unaccomplished tasks. The knowledge base helps the robot taking optimal or good decisions for any new coming mission. This introduces another layer of intelligence, allowing the robots to better coordinate to efficiently execute the tasks. For example, if a robot is known to have failed accomplishing some previously assigned tasks, it can be excluded from coordination on future similar tasks.

On the other hand, in addition to common software engineering requirements namely modularity, extensibility and reuse, we considered the following four key non-functional requirements for the development of the software architecture.

- **Decentralization:** any algorithm or protocol to be implemented must be fully distributed in the sense that any application should not rely on a central agent (or unit) for taking decisions, but decisions should be taken by robots with the partial knowledge they would have about the system. This is a core requirement, as it is not realistic to assume that all robots are fully connected all the time with each other or with a central system, as they are typically dotted with limited range wireless transceivers.
- **Fault-Tolerance:** any application should continue its operation correctly even if some robots become faulty before/during/after the execution of any mission. This is indeed a natural consequence from the aforementioned decentralization requirement, which avoids the single point of failure problem. Indeed, if any robot fails or has its battery deplete, other robots should detect this and should manage to undertake the task previously assigned to the dead robot.
- **Performance measure:** it is important to be able to evaluate how were successful the set of robots to achieve the overall mission. Several criteria could be used for that purpose, some of them may depends on the application. The criterion may vary from the satisfactory behavior to an optimal behavior.
- **Heterogeneity:** In multi-robot systems, it is typical to deploy robots of different types and brands. One of the major requirements of our architecture is to support different types of robots, so that the same code be reused by different types of robots without major changes. This is an important issue to ensure modularity and code reuse.

4.2 Software Architecture for ROS-enabled Cooperative Robots

General Overview In this section, we present our multi-robot software architecture and we demonstrate how it is integrated with ROS. The COROS architecture is based on the concept of multi-agent, where an agent represents an independent entity, typically a robot machine (i.e. Robot Agent) or a monitoring or control workstation (i.e. Monitor Agent). Unless otherwise specified, by default we consider an *agent* as a Robot Agent. Each agent is composed of a set of *components* that build the internal behaviour of the robot and allow it to interact remotely with other agents. In what follows, we present the main components of the software architecture and explain how they satisfy the aforementioned requirements.

Architecture Components Figure 2 shows the component diagram of the software architecture. The software system is decomposed into five subsystems, each of which plays the role of a container of a set of components. These subsystems are:

1. **Communication:** this subsystem was designed to address communication requirement, as it ensures the inter-robot interaction between different agents. It comprises extensible and modular client and server components that enable agents

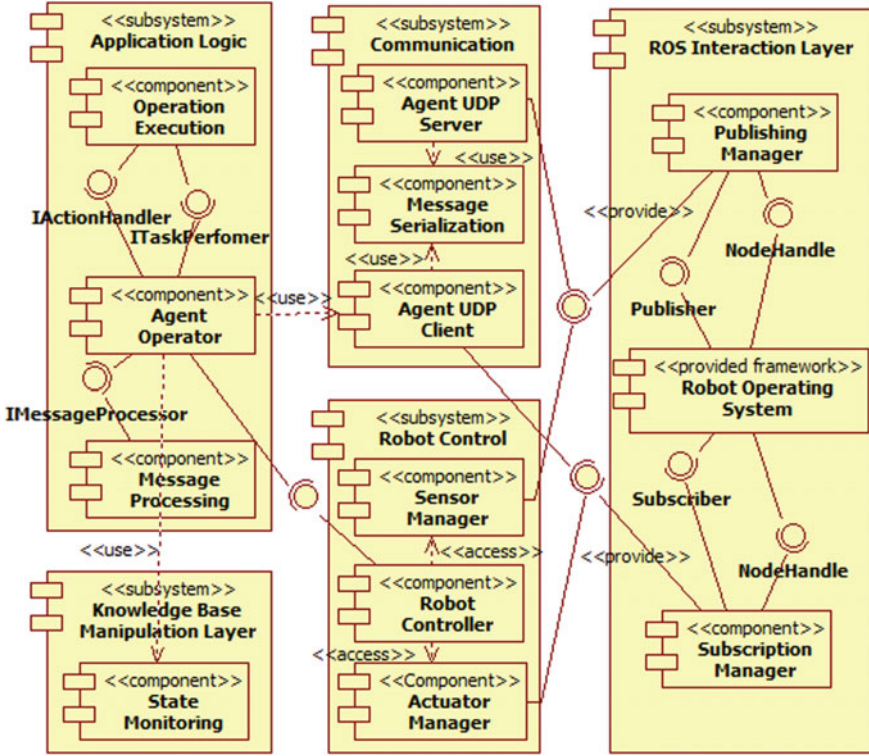


Fig. 2 COROS components

to exchange serialized messages through the network interface using sockets. One major challenge encountered is the incompatibility between socket-based messages and ROS messages. Indeed, as mentioned in Sect. 2, message-passing between nodes in ROS requires a particular message structure that is different from the structure of message received from/send through network interfaces. For that purpose, we designed a new component for message processing which maps any message between ROS and network interfaces. Indeed, any message received through sockets is converted to a ROS-compatible message and vice-versa.

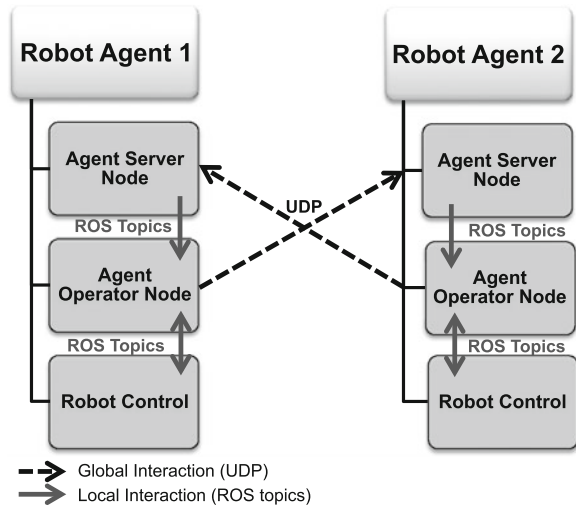
- 2. **ROS Interaction Layer:** this subsystem adds a lightweight layer on top of ROS allowing a seamless inter-process interaction between ROS nodes (processes) defined in the architecture. The main role of this layer is to provide a simple and efficient way to manage the subscribers and the publishers to ROS topics and services. Any node can publish or subscribe to a new topic using both components *PublishingManager* and *SubscriptionManager* without having to directly interact with ROS. These components will use the *NodeHandle* object, needed to create any ROS node, of the class that uses them to publish or subscribe to topics.

These components define (*key, value*) map data structures to manage topics to be published or subscribed, where each topic is identified by a unique key in the map. This key is used to as topic reference to publish or subscribe to any topic. ROS services are also supported in the same way.

3. **Robot Control:** this subsystem also adds a second layer on top of ROS providing a bridge between the local software agents and the physical robots. The role of this layer is to manage the robot configuration and its state. The Robot Controller component provides an abstraction model for any ROS-enabled robot. Indeed, this component provides several interfaces for controlling and monitoring robots states such as location, published and subscribed topics, provided and used services, etc. This enables to make easier to management of heterogeneous robots as they adhere to a common component model. Any robot type can be easily configured to provide the interfaces provided by the robot controller components.
4. **Application Logic:** this subsystem addresses the problem solving requirements; it encapsulates all of the components needed to implement a complete multi-robot application. Any new application should reuse and configure the software components to define its proper behaviour. The *Agent Operator* is the main component of the Application Logic subsystem as it implements the actual behaviour of the applications. This means that every type of received message (through Agent Server Component) triggers the execution of an appropriate function as specified by the application. The Agent Operator uses the Communication subsystem to exchange information with other robotic agents in the environment. As mentioned above, the message processor is used to provide a mapping between Socket-based messages and ROS messages, and make message serialization/deserialization upon sending through/receiving from network interfaces. For the execution of a task (e.g. moving to a certain location), the Agent Operator interact with the Operation Execution component through two possible interfaces: (1) the Task Performer interface (ITaskPerformer) or (2) the Action Handler interface (IActionHandler).
5. **Knowledge Base Manipulation Layer:** This subsystem aims at satisfying knowledge base requirement and maintain an up-to-date information about the robot status and its environment. Currently, we did not use a specific formal language either for knowledge representation or reasoning in this subsystem. This in reality related to the nature of the service Robots applications, when each robot gather the captured information from its environment to accomplish a specific task. Usually, the majority of gathered information becomes obsolete from an execution to another. Based on its unique component *State Monitoring*, this subsystem provides to others with a useful information and services such as allowing the agent to monitor and control its local state, other agents' states, the state of the different tasks, and the information about the agent initial configuration.

Notice that in the architecture of Fig. 2, there is no direct link, association or interface between the Client and the Server, or between the Robot Control components. Indeed, the communication between these two components is ensured by ROS topics in the case of local (inter-process) interactions, and by the UDP protocol in the case

Fig. 3 Global and local interactions



of global (external) interactions. Figure 3 illustrates the interaction model between two robotic agents. In this figure, it is noted that the Client is not explicitly shown as the Agent Operator uses the Agent UDP Client component in a way that it extends all of its functionalities. Indeed, the Agent Operator sends network messages through the UDP Client Component.

The Agent UDP Server is a component that ensures the reception of messages through network interfaces using UDP sockets. This Agent UDP Server receives messages from any agent communicating through a specified application port. When a message is received, the Agent UDP Server forwards it to the Agent Operator through the ROS topic publishing mechanism. As a central component, the Agent Operator should be subscribed to any message topic published by the Agent UDP Server, and then makes call to other interfaces from different components (e.g. *IMessageProcessor*, *ITaskPerformer*, and *IActionHandler*), when instantiated for a specific application, to define the logic of the underlying application.

Implementation Classes Figure 4 illustrates the Class Diagram that presents the detailed structure of each aforementioned subsystem, in addition to the relationships between classes. In what follows, we present some important points about the implemented classes and technical choices.

In the class diagram of Fig. 4, the five subsystems are depicted and they are interpreted as logic containers (i.e. doesn't exist at the file system level) of a set of implementation packages. At the file system level, we identify *packages* that are represented by separate folders, each of which represents either a part of a component (monitoring and state), a full component (processing, execution, operator, robot, sensors, and actuators), or two components or more (core). Physically, classes are embedded in their respective packages.

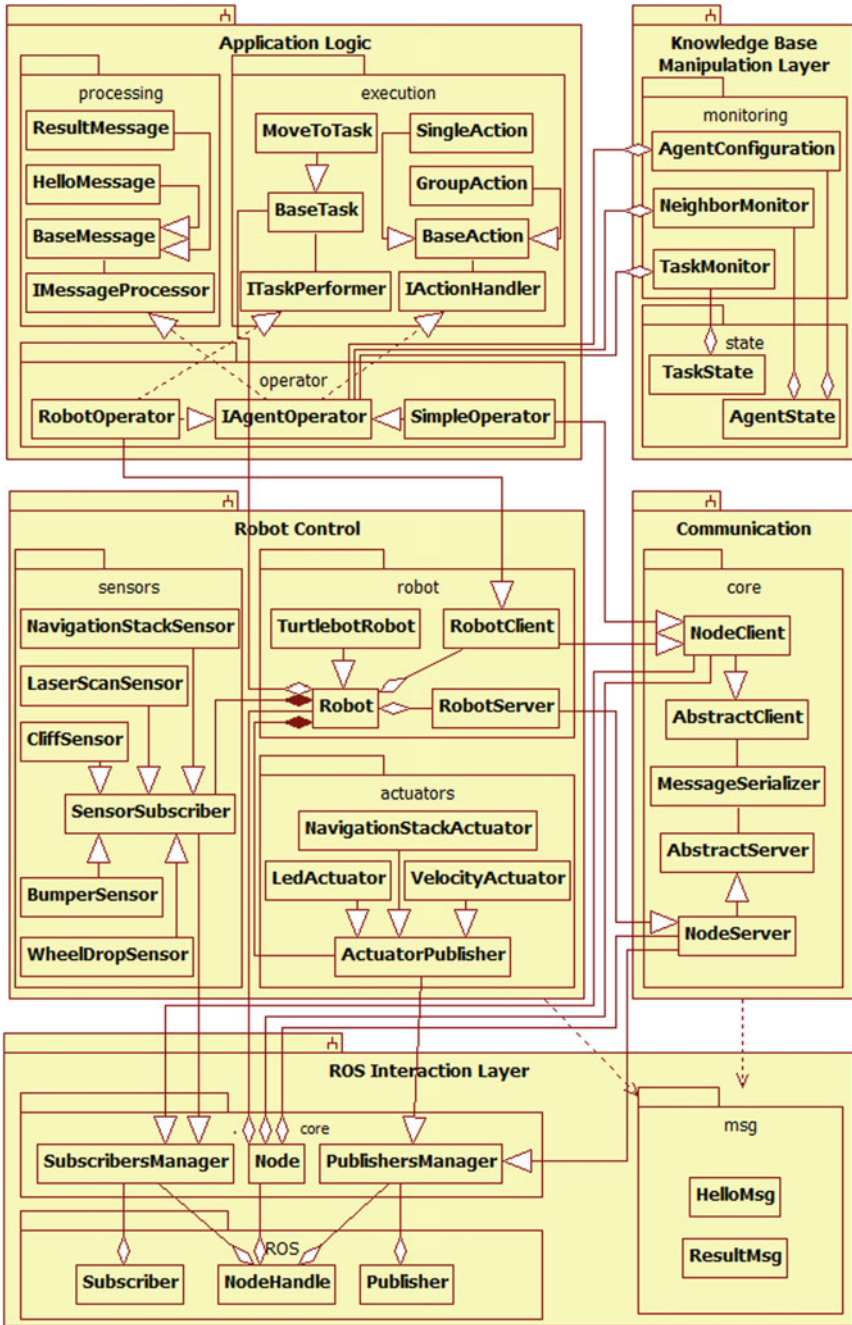


Fig. 4 COROS main classes

Table 1 COROS implementation packages usage

Subsystem	Package	Usage
ROS interaction layer	ROS	Refers to the (non-implemented, but) imported ROS library classes
	core	Simplifies the interaction between multi-robot application and ROS framework
	msg	Folder, container of ROS primitive message files (structure of messages used to publish in ROS topics)
Communication	core	Enables client/server communication using UDP protocol (different agents), client/server communication using ROS topics (same agent), and message serialization
Robot control	robot	Provides access to real robot properties, sensors and actuators. Also, enables client/server accessing robot functionalities
	sensors	Simplifies handling sensors topics subscription
	actuators	Simplifies handling actuators topics publishing
Knowledge base manipulation stack	state	Stores information about agents and tasks
	monitoring	Manages configuration about an agent and monitors agents and tasks states
Application logic	processing	Provides an interface and a base class for processing messages
	execution	Provides interfaces and base classes for handling actions and performing tasks
	operator	Concatenates the processing and execution provided interfaces

Tables 1 and 2 briefly describe the COROS implemented packages and classes, respectively. The objective of COROS implementation architecture (i.e. packages and classes) is to provide the robotic software developer with a set of reusable classes, that can be effectively and easily used/extended for developing new applications, which help speeding up the implementation process, and promoting the code reusability.

There are four major types of classes to be used for the development of any new application: (1) server (*NodeServer* and *RobotServer*), (2) operator (*SimpleOperator* and *RobotOperator*), (3) operation (*BaseAction* and *BaseTask*), and (4) message (*BaseMessage*). When developing a new application, the software developer needs to identify the agents composing the application and their different roles. According to the role of an agent, the choice of the server and the operator is based on whether to interact with a robot (i.e. *RobotServer* and *RobotOperator*) or not (i.e. *NodeServer* and *SimpleOperator*). Similarly, the choice of operations to be implemented is based on whether the operation involves an actuator (the operation is called *Task*), such as robot wheel motors, or not (the operation is called *Action*). To ensure code reuse, classes of type *Task*, *Action*, or *Message* should be derived

Table 2 COROS main classes description

Subsystem	Package	Class	Description
ROS interaction layer	core	Node	Encapsulates the NodeHandle. Aggregated by classes who need to communicate with ROS topics mechanism
		PublishersManager	Manages several ROS publishers using a single NodeHandle. Aggregated or extended by classes who need to create different topics' publishers and publish messages
		SubscribersManager	Manages several ROS subscribers using a single NodeHandle. Aggregated or extended by classes who need to create different topics' subscribers and receive messages
Communication	core	AbstractServer	Binds a UDP socket on a specified port and receives and deserializes messages
		AbstractClient	Serializes and sends or broadcasts messages using UDP protocol
		NodeServer	Enhances the UDP server with ROS capabilities (namely: publishing messages)
		NodeClient	Enhances the UDP client with ROS capabilities (namely: subscribing to topics)
		MessageSerializer	Provides serialization/deserialization for client and server
Robot control	robot	Robot	Represents a real robot with its properties and allows access to all robots sensors and actuators
		Turtlebot	Represents the Turtlebot robot, its properties and the needed sensor subscriptions and actuator publishing
		RobotServer	Enhances a ROS enabled UDP server to access to robot functionalities
		RobotClient	Enhances a ROS enabled UDP client to access to robot functionalities
	sensors	SensorSubscriber	Simplifies handling sensors topics subscription
		LaserScanSensor	Provides subscription to Laser Scan sensor captured data
	actuators	ActuatorPublisher	Simplifies handling actuators topics publishing
		NavigationStack Actuator	Allows publishing Navigation Stack data

(continued)

Table 2 (continued)

Subsystem	Package	Class	Description
Knowledge base manipulation stack	state	AgentState	Stores information about an agent (its ID, status, role, etc.)
		TaskState	Stores information about a task (its ID, status, performed by, etc.)
	monitoring	Agent-Configuration	Manages agent configuration information and state
		NeighborMonitor	Gathers information about the state of neighbor agents
		TaskMonitor	Gathers information about the state of tasks
	Application logic	processing	IMessage-Processor
BaseMessage			Base class for all application related messages
HelloMessage			A message class to be used for announcing agent information to its neighbors
ResultMessage			A message class to be used for announcing agent task end
execution		IActionHandler	Interface, to be implemented by classes need to handle actions
		BaseAction	Base class for all application related actions
		SingleAction	Base class for all application related single actions; which execute once
		GroupAction	Base class for all application related group actions; which execute several times and save the context between different executions
		ITaskPerformer	Interface, to be implemented by classes need to perform tasks
		BaseTask	Base class for all application related tasks
		MoveToTask	A sample class that can be used as base class for tasks need to make the robot moving to a specific target
operator		IAgentOperator	Interface, concatenates the methods of IMessageProcessor and IActionHandler and realizes some of them
		SimpleOperator	Extends NodeClient and realizes IAgentOperator to provide a client class capable to communicate using UDP and ROS topics, to process messages, and to handle actions
		RobotOperator	Extends NodeClient and realizes IAgentOperator and ITaskPerformer to provide a client class capable to communicate using UDP and ROS topics, to process messages, to handle actions, and to perform robot tasks

from *BaseTask*, *BaseAction*, and *BaseMessage*, respectively. With respect to the *Message* class instantiation, the implemented class should define a member function facilitating the message serialization. Another approach would be to encode any message as a string in the JSON format.

Conversion Between Application Messages and ROS Messages As mentioned above, there are two types of communications: (1) The communication between agents, carried out by the UDP protocol by using serialized messages, referred to as *Application Messages*, which is typically derived from *BaseMessage* class, and (2) communication between processes (ROS nodes) of a single agent referred to as inter-process communication and is carried out by *ROS Messages*. Once an agent receives an application message from its UDP interface, it must forward this received message to ROS system for processing it and performs required actions and/or tasks. For that purpose, we have developed a conversion between ROS Messages and Application Messages and vice-versa. To illustrate this through an example, consider the following ROS message structure used for the discovery application that allow neighbor agents to discover each other (refer to Sect. 5.1 for details about the application).

```
int32 message_code # the code designating the needed operation
int32 agent_id     # the ID of the agent
string agent_ip    # the IP address of agent's machine
int32 agent_port   # the port number of agent's machine
string agent_role  # the role played by an agent in the app.
string agent_status # the current status of the agent
float64 timestamp  # the timestamp of the last status
```

At compilation time, ROS system auto-generates C++ header files corresponding to this ROS message structure. This ROS message will have an equivalent serialized Application Message. The idea of serialization is the simplest way to define a standard UDP client and server since the size of a string variable is easily computable, but is not the case to compute the size of the instance of a class. For that purpose, Boost Serialization¹ library were the most stable and best fit our need. The equivalent serialized application message is the following:

```
template<class Archive>
void DiscoveryMessage::serialize(Archive & ar,
    const unsigned int version){
    ar & message_code;
    ar & id;
    ar & ip;
    ar & port;
    ar & role;
    ar & status;
    ar & timestamp;
}
```

¹http://www.boost.org/doc/libs/1_55_0/libs/serialization/doc/index.html.

4.3 Guidelines for Developing New Applications

In what follows, we provide a simple methodology that guides the robotic software developer in developing its own application using the COROS architecture. Indeed, the main goal behind proposing COROS architecture is to provide the main building blocks needed to develop distributed multi-robot applications favoring the cooperation between ROS-enabled robots. We define two main phases for the development of new applications:

1. The Design Phase

- (a) Determine how many agents (and their roles, e.g.: robot, monitor, etc.) are needed for the application.
- (b) Define the application message types and their structure.
- (c) Define ROS topics and messages files with their contents.
- (d) Design operations (actions and/or tasks) to process application messages.
- (e) Design ROS callback functions (subscriber side) to process ROS messages.
- (f) Determine addition information for the configuration of an agent (this is done only in the case of the standard provided configuration structure (*AgentConfiguration* class) does not suffice).

2. The Implementation Phase

- (a) Implement the application-specific configuration class (if any) as subclass of *AgentConfiguration*.
- (b) Create *ROSmessage* files according to ROS specific format.
- (c) Create *Applicationmessage* classes (subclass of *BaseMessage*) and implement serialization functions.
- (d) Create action classes (subclass of *BaseAction*) and task classes (subclass of *BaseTask*), then override their respective *execute()* functions.
- (e) Create a server class as a subclass of *NodeServer* or *RobotServer*, according to the agent role, for each agent by redefining their constructors, overriding the initialization methods *init()* for creating publishers of the specified topics, and overriding the *forward()* method for deserializing received application messages and publishing them as ROS messages to the operator.
- (f) Create an operator class as a subclass of *SimpleOperator* or *Robot-Operator*, according to the agent role, for each agent by redefining the constructor, and implementing *init()* method for creating subscribers to the specified topics, and the callback methods related to each topic subscription for message processing.

5 Experimental Validation

In what follows, we present two experimental application scenarios to validate COROS architectural concepts and to demonstrate how it is effective in building new distributed robotic applications while ensuring component reuse and extensibility.

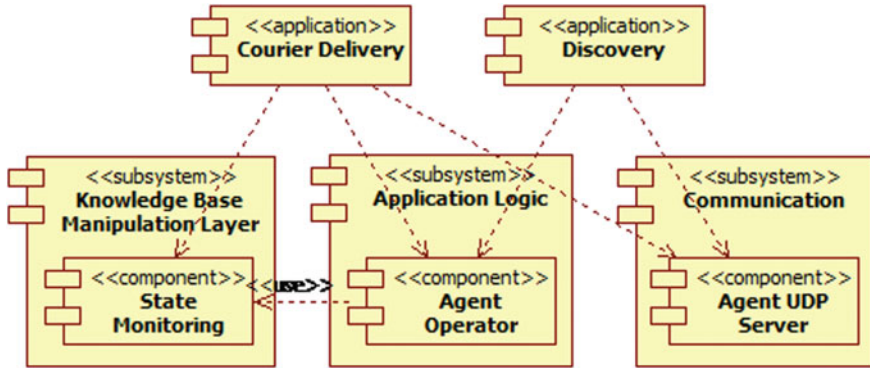


Fig. 5 Discovery and courier delivery applications in COROS architecture

The documentation and source code of COROS are available in [36]. The first scenario describes a discovery protocol which allows agents to discover their neighborhood. The second scenario describes a courier delivery from an office to another. Figure 5 depicts the different subsystems directly used by these two applications. The remaining subsystems are also used but indirectly.

5.1 Scenario 1: Discovery Protocol

Overview: In this section, we present the discovery protocol as an application that allows robots to discover their neighborhood, using the COROS architectural concepts. The reason behind implementing the discovery protocol is because it is needed by typical distributed robotic applications, since a robot basically needs to have information about its neighbors to be able cooperate with them in accomplishing missions. We build the discovery protocol as an independent ROS software component that can be used by any other ROS application.

The discovery protocol is simple. Each robot periodically sends a *HELLO* message carrying out information needed to identify a robot. Once a robot receives a *HELLO* message from a neighbor robot, it adds it to its neighbor list. The timestamp field allows to keep track of the last *HELLO* message update from a neighbor robot and can be used to discard robots with outdated record, so that to maintain the neighborhood list up-to-date when robots are moving.

Figure 6 presents the classes and components reused to implement the discovery application based on COROS architecture.

Implementation: As depicted in Fig. 6, the discovery package comprises two main classes: (1) The *DiscoveryOperator* is a class, which maps to an *AgentOperator* component in the COROS architecture, and that inherits from the *SimpleOperator* class providing all interfaces for processing any incoming message according to discovery application logic. For each received message, the robot will update its neighborhood

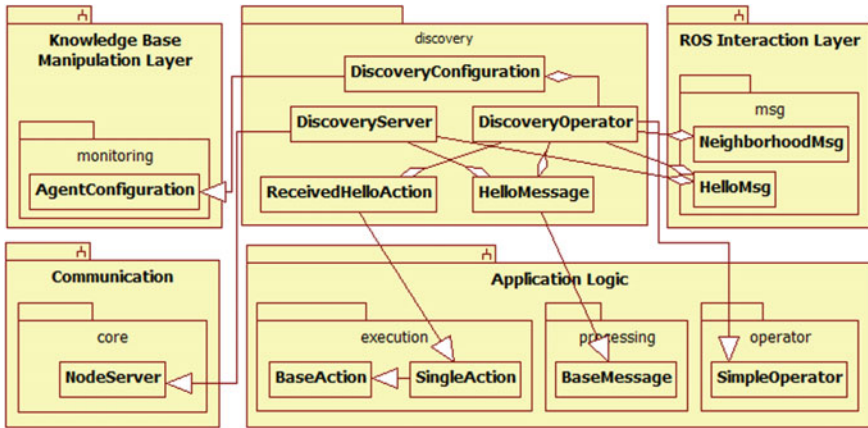


Fig. 6 Discovery application class diagram

table and publishes it as a *NeighborhoodMsg* topic which can be used by any ROS node in the robot. For example, in a bidding-auctioning process, a robot may need to contact its neighbor robots to bid on a certain task. (2) The *DiscoveryServer* is responsible for handling network communication between robots, by sending/receiving message to/from other robots. Received applications messages are decoded in the server class and forwarded as *HelloMsg* ROS messages to the *DiscoveryOperator*, which processes the message and executes the action accordingly, as explained above.

Discussion: The COROS architectural components and classes enabled us to easily implement the discovery application as a new ROS component that can be used by any other ROS node (i.e. process) that requires information about neighbor robots. Indeed, the discovery application can be executed as a set of ROS nodes, i.e. *DiscoveryServer* and *DiscoveryOperator*, that can be easily configured in a ROS launch file, taking as parameters the *RobotID*, the port number of the discovery application, the robot IP address and the frequency of *HELLO* messages. The discovery application updates the neighborhood table of the robot and publishes this table as *robot_id/neighborhood_list* topic that can be used by any ROS node by subscribing to this topic. Figure 7 depicts an example of execution of the discovery protocol in a robot and the *robot1/neighborhood_list* published topic. In this example, only one robot with ID equal to 1 is discovered in the neighborhood.

5.2 Scenario 2: Courier Delivery Application

Overview: In this section, we present the courier delivery application to test a more advanced scenario that involves cooperation between a team of robots and to demonstrate how COROS architecture is effective in building such distributed

```

/home/akoubaa/catkin_ws/src/robot_controller/launch/discovery.launch http://localhost:
timestamp: 1.4051e+09
.
[ INFO] [1405098830.397059267]:
[Discovery Robot Client]
[Event]: received HELLO message from a robot:
message_code: 200
id: 1
ip: 192.168.1.108
port: 25000
timestamp: 1.4051e+09
.
[Action]: Add robot to robot neighbor list

Neighborhood Table
|-----|
| ID | IP Address      | Port   | Timestamp |
|-----|
| 1  | 192.168.1.108  | 25000  | 1405098830 |
|-----|
[ INFO] [1405098830.398286145]: Publisher "neighborhood_pub" published a message
akoubaa@anis-vbox:~$ rostopic echo /robot1/neighborhood_list
robot_ip_addresses: ['192.168.1.108']
timestamp: [1405098910.392974]
---
robot_ip_addresses: ['192.168.1.108']
timestamp: [1405098913.3972328]
---
robot ip addresses: ['192.168.1.108']

```

Fig. 7 Discovery protocol published topic and execution

applications. We chose the courier delivery application as a proof-of-concept of the COROS architecture and its implementation.

We consider a team of N robots receive a command for delivering a courier from one office to another, coordinate together using a market-based mechanism to elect the robot with the lowest cost to execute the task. The scenario is as follows: A user sends a courier delivery mission as an auction to the team of robot. The auction message includes the location of the two offices, the sender of the courier and the receiver. Each robot receiving the courier delivery task calculates its bid for executing the action and sends the bid to the auctioneer. The bid represents an estimate of the total distance that the robot has to travel between the two offices starting from its location. Once all bids are received, the auctioneer selects the robot with the lowest bid to execution the mission. The winning robot moves to the sender office first to get the courier and then goes to the receiving office for delivery. We have deployed this application in the Computer Science department at Prince Sultan University using two robots and four offices as illustrated in Fig. 8. A map of the environment was established using ROS gmapping package. A video demonstration explaining the scenario is available in the iroboapp project page [37]. In this scenario, the robot with the lowest cost was select and has successfully delivered the courier from one office to another.



Fig. 8 Courier delivery deployment environment

Implementation: As depicted in Fig. 9, the courier delivery application package is implemented following COROS architectural concepts. The *Communication*, *ROS Interaction Layer*, *RobotControl*, *Application Logic* and *Knowledge Base Manipulation Layer* packages provide the required classes and components for the *courier* package that encodes the behavior of the application.

The courier packages comprises the following main modules: (1) a user module that includes a *CourierUserServer* a subclass of *NodeServer* and *CourierUserOperator* a subclass of *SimpleOperator* that allow an end-user to send mission to the robots and process received bids, respectively, (2) a robot module that includes *CourierRobotServer* a subclass of *RobotServer* and *CourierRobotOperator* a subclass of *RobotOperator* allowing a robot to receive mission orders and process commands. The user and robots exchange different types of messages depending on the type of interaction of the market-based protocol.

The execution of the delivery mission is ensure by the *CourierDeliveryTask* class, which reuses the *MoveToTask* generic class of the *execution* package to define how the robot should execute the mission, which in this case, moving to the sender office, then moving to the receiving office after getting the courier. The *CourierDeliveryTask* uses an instance of the *TurtlebotRobot* needed to have access to all sensor and actuator of the Turtlebot robot used in the experiments. It will be straightforward to use the same code on another type of robot, which just requires to change the type of robot instance to the appropriate one. Of course, it is needed to develop the Robot Control module for any new robot to be used in the COROS framework.

Discussion: Figure 10 depicts the deployment diagram of the courier application. It is clear that the COROS architecture facilitates us the development of the courier

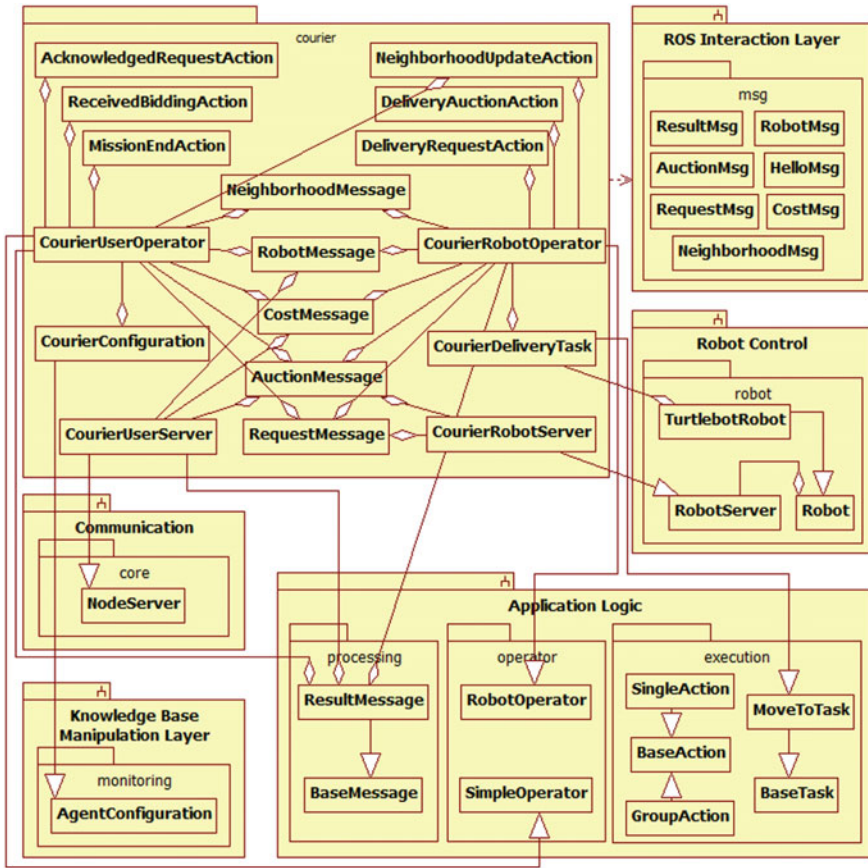


Fig. 9 Courier delivery implemented classes

application that involve both an end-user and a robot, each has its own server and operator components, as depicted in Fig. 10. It can be observed that the Discovery application was used as an independent component of the COROS framework in both the user and robot machines. Adding any other application in the COROS framework is easy and any application can interact with any other using the ROS middleware through ROS topics and services. In addition, all components can be configured using a ROS launch files.

The major advantage of the COROS architecture is that it allows building new distributed robotic applications by instantiating existing generic COROS classes and packages and making abstraction to several low-level implementation (e.g. network communication, ROS primitives) details embedded in these generic packages.

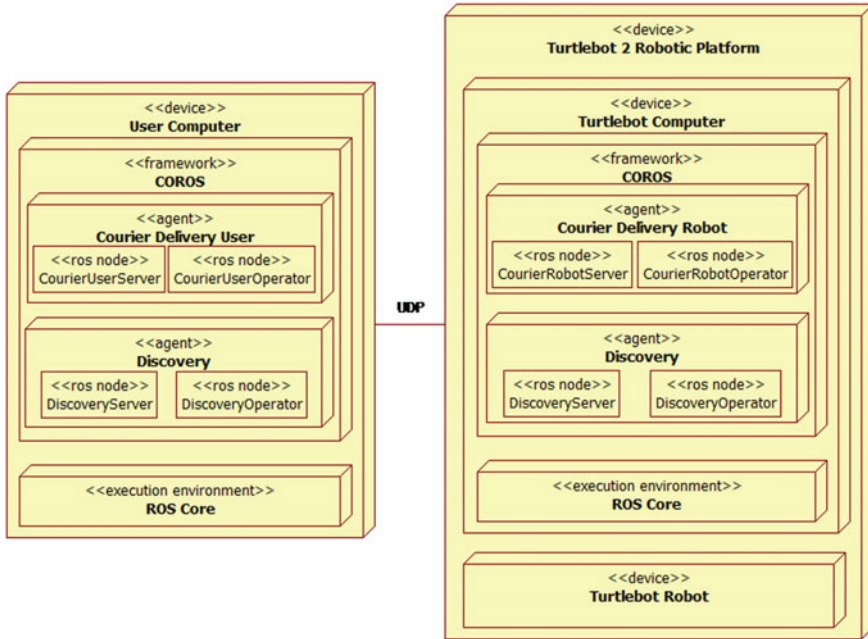


Fig. 10 Courier delivery application deployment based on COROS implementation

6 Conclusion and Future Works

In this chapter, we proposed COROS, a multi-agent software architecture for cooperative service robots applications. At conceptual level, COROS comprises five sub-systems including communication, ROS interaction layer, robot control, application logic and knowledge bases. Each system provides an abstraction layer to the robotics software developer that allow to facilitate the design and implementation of new distributed robotic applications. The COROS framework has the advantage of promoting software reuse and modularity through the adoption of a component-based approach in the design of the architecture. The COROS architecture was extensively validated through two distributed applications including the discovery protocol and the courier delivery application in indoor environments. It has been demonstrated how COROS was effective in building these two applications.

Although the COROS framework was proven to be effective, we are planning several extensions for incorporating advanced functionalities. First, we have the objective of contributing to promote COROS to support Rapid Application Development (RAD) in robotic context by providing utilities to automate the creation of new agent servers and operators for any new application instead of editing them manually. Another extension consists in improving and standardizing message serialization. Currently, messages are serialized using the *boost* library and we aim at serialization

message into JSON or ROS format so that it becomes platform-independent. Furthermore, we aim at hiding more network complexity by adding an abstraction layer that allows the application developer to deal only with ROS topic without the need to handle network and message serialization issues.

We believe that COROS provides an important milestone in the design and development of distributed robotic application on top of the ROS middleware.

Acknowledgments This work is supported by the iroboapp project “Design and Analysis of Intelligent Algorithms for Robotic Problems and Applications” under the grant of the National Plan for Sciences, Technology and Innovation (NPSTI), managed by the Science and Technology Unit of Al-Imam Mohamed bin Saud University and by King AbdulAziz Center for Science and Technology (KACST). This work is also supported by the myBot project entitled “MyBot: A Personal Assistant Robot Case Study for Elderly People Care” under the grant from King AbdulAziz City for Science and Technology (KACST). This work is partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme ‘Thematic Factors of Competitiveness’), within project FCOMP-01-0124-FEDER-037281 (CISTER); by Prince Sultan University.

References

1. World Robotics 2013 Service Robots: <http://www.ifr.org/service-robots/statistics/> (2013)
2. The Player/Stage Project: <http://playerstage.sourceforge.net/>
3. The MIRA Project: <http://www.mira-project.org/>
4. Mobile Robot Programming Toolkit (MRPT): <http://www.mrpt.org/>
5. Robot Operating System (ROS): <http://www.ros.org>
6. Westhoff, D., Zhang, J.: A unified Robotic software architecture for service Robotics and networks of smart sensors. In: Berns, K., Luksch, T. (eds.) *Autonome Mobile Systeme 2007*, Informatik Aktuell, pp. 126–132. Springer, Berlin (2007)
7. Kim, M., Kim, S., Park, S., Choi, M.-T., Kim, M., Gomaa, H.: UML-based service robot software development: a case study. In: *Proceedings of the 28th International Conference on Software Engineering, ICSE’06*, pp. 534–543. ACM, New York, USA (2006). doi:10.1145/1134285.1134360, <http://doi.acm.org/10.1145/1134285.1134360>
8. Kim, M., Kim, S., Park, S., Choi, M.-T., Kim, M., Gomaa, H.: UML-based service robot software development: a case study. In: *Advances in Service Robotics, InTech*, pp. 127–148 (2008).doi:10.5772/5947
9. Wojtczyk, M.: A new model to design software architectures for mobile service robots, Dissertation, Technische Universität München, München, Germany (2010)
10. Viguria, A., Maza, I., Ollero, A.: Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions. *Adv. Robot.* **24**(1–2), 1–23 (2010)
11. The Iroboapp Project: <http://www.iroboapp.org> (2014)
12. O’Kane, J.M.: A gentle introduction to ROS, independently published (2013). <http://www.cse.sc.edu/jokane/agitr/>
13. Wyrobek, K., Berger, E., Van der Loos, H., Salisbury, J.: Towards a personal Robotics development platform: rationale and design of an intrinsically safe personal Robot. In: *IEEE International Conference on Robotics and Automation, ICRA 2008*, pp. 2165–2170. IEEE (2008)
14. Quigley, M., Berger, E., Ng, A.Y.: Stair: hardware and software architecture. In: *AAAI 2007 Robotics Workshop*, Vancouver, BC, pp. 31–37 (2007)
15. Tardioli, D.: Real-time communication in wireless Ad-Hoc networks. The RT-WMP Protocol, Ph.D. thesis, Universidad de Zaragoza (October 2010)

16. Brugali, D., Prassler, E.: Software engineering for Robotics [from the guest editors]. *IEEE Robot. Autom. Mag.* **16**(1), 9–15 (2009)
17. Calisi, D., Censi, A., Iocchi, L., Nardi, D.: Design choices for modular and flexible Robotic software development: the OpenRDK viewpoint. *J. Softw. Eng. Robot.* **3**(1), 13–27 (2012)
18. Brooks, A., Kaupp, T., Makarenko, A., Williams, S., Oreback, A.: Towards component-based Robotics. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS 2005), pp. 163–168. IEEE (2005)
19. Lee, T.-Y., Seo, H.-R., Lee, B.-H., Shin, D.-R.: A software component model and middleware architecture for intelligent mobile Robot. In: 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), vol. 4, pp. 453–456. IEEE (2010)
20. Hasselbring, W.: Component-Based Software Engineering. *Handbook of Software Engineering and Knowledge Engineering*
21. Broten, G., Mackay, D., Monckton, S., Collier, J.: The robotics experience. *IEEE Robot. Autom. Mag.* **16**(1), 46–54 (2009). doi:[10.1109/MRA.2008.931632](https://doi.org/10.1109/MRA.2008.931632)
22. Iborra, A., Caceres, D., Ortiz, F., Franco, J., Palma, P., Alvarez, B.: Design of service robots. *IEEE Robot. Autom. Mag.* **16**(1), 24–33 (2009)
23. Lau, K.-K., Wang, Z.: Software component models. *IEEE Trans. Softw. Eng.* **33**(10), 709–724 (2007)
24. Bruyninckx, H.: Open Robot control software: the OROCOS project. In: IEEE International Conference on Robotics and Automation. Proceedings 2001 ICRA, vol. 3, pp. 2523–2528. IEEE (2001)
25. Lütkebohle, I., Philippsen, R., Pradeep, V., Marder-Eppstein, E., Wachsmuth, S.: Generic middleware support for coordinating Robot software components: the task-state-pattern. *J. Softw. Eng. Robot.* **2**(1), 20–39 (2011)
26. Angerer, A., Hoffmann, A., Schierl, A., Vistein, M., Reif, W.: Robotics API: object-oriented software development for industrial Robots. *J. Softw. Eng. Robot.* **4**(1), 1–22 (2013)
27. Limbu, D., Tan, Y.-K., Jiang, R., Dung, T.A.: A software architecture framework for service Robots. In: 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1736–1741. doi:[10.1109/ROBIO.2011.6181540](https://doi.org/10.1109/ROBIO.2011.6181540) (2011)
28. Luzzana, A.: Classification and integration of software component models for Robotics, Ph.D. thesis, Università degli studi di Bergamo (April 2013)
29. DeLoach, S.A., Matson, E.T., Li, Y.: Exploiting agent oriented software engineering in cooperative Robotics search and rescue. *Int. J. Pattern Recognit. Artif. Intell.* **17**(05), 817–835 (2003)
30. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent systems engineering. *Int. J. Softw. Eng. Knowl. Eng.* **11**(03), 231–258 (2001)
31. Parker, L.E.: Current state of the art in distributed autonomous mobile robotics. In: *Distributed Autonomous Robotic Systems*, vol. 4, pp. 3–12. Springer, Heidelberg (2000)
32. Matson, E., DeLoach, S.: Enabling intra-robotic capabilities adaptation using an organization-based multiagent system. In: 2004 IEEE International Conference on Robotics and Automation. Proceedings. ICRA'04, vol. 3, pp. 2135–2140. doi:[10.1109/ROBOT.2004.1307378](https://doi.org/10.1109/ROBOT.2004.1307378) (2004)
33. Silva, D., Braga, R.A.M., Reis, L., Oliveira, E.: A generic model for a Robotic agent system using GAIA methodology: two distinct implementations. In: 2010 IEEE Conference on Robotics Automation and Mechatronics (RAM), pp. 280–285. doi:[10.1109/RAMECH.2010.5513176](https://doi.org/10.1109/RAMECH.2010.5513176) (2010)
34. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. *Auton. Agent. Multi-Agent Syst.* **3**(3), 285–312 (2000)
35. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: the Gaia methodology. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **12**(3), 317–370 (2003)
36. COROS: <http://www.iroboapp.org/index.php?title=COROS> (2014)
37. Office courier delivery application. video demonstration, iroboapp project. <http://www.iroboapp.org/index.php?title=Videos> (2014)

Multi-robot Task Allocation: A Review of the State-of-the-Art

Alaa Khamis, Ahmed Hussein and Ahmed Elmogy

Abstract Multi-robot systems (MRS) are a group of robots that are designed aiming to perform some collective behavior. By this collective behavior, some goals that are impossible for a single robot to achieve become feasible and attainable. There are several foreseen benefits of MRS compared to single robot systems such as the increased ability to resolve task complexity, increasing performance, reliability and simplicity in design. These benefits have attracted many researchers from academia and industry to investigate how to design and develop robust versatile MRS by solving a number of challenging problems such as complex task allocation, group formation, cooperative object detection and tracking, communication relaying and self-organization to name just a few. One of the most challenging problems of MRS is how to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints. This problem is known as Multi-robot Task Allocation (MRTA) problem. MRTA is a complex problem especially when it comes to heterogeneous unreliable robots equipped with different capabilities that are required to perform various tasks with different requirements and constraints in an optimal way. This chapter provides a comprehensive review on challenging aspects of MRTA problem, recent approaches to tackle this problem and the future directions.

A. Khamis (✉)

Engineering Science Department, Suez University Egypt and Vestec. Inc.,
125 Northfield Drive West, Waterloo N2L 6N8, Canada
e-mail: akhamis@pami.uwaterloo.ca

A. Hussein

Intelligent Systems Lab (LSI) Research Group, Universidad Carlos III de Madrid (UC3M),
Av. de la Universidad 30, 28911 Madrid, Spain
e-mail: ahussein@ing.uc3m.es

A. Elmogy

Computers and Control Engineering Department, Tanta University,
Egypt and Arab East Colleges, Abdalla Elsayhmi Road, Elhada District, Riyadh 11583, KSA
e-mail: amelmoghy@arabeast.edu.sa

© Springer International Publishing Switzerland 2015

A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604, DOI 10.1007/978-3-319-18299-5_2

1 Introduction

Multi-robot systems (MRS) are a group of robots that are designed aiming to perform some collective behavior. By this collective behavior, some goals that are impossible for a single robot to achieve become feasible and attainable. MRS have been on the agenda of the robotics community for several years. It is only in the last decade, however, that the topic has really taken off, as seen from the growing number of publications appearing in the journals and conferences. One of the reasons that the topic has become more popular is the various foreseen benefits of MRS compared to single robot systems. These benefits include, but are not limited to the following:

- **Resolving task complexity:** some tasks may be quite complex for a single robot to do or even it might be impossible. This complexity may be also due to the distributed nature of the tasks and/or the diversity of the tasks in terms of different requirements.
- **Increasing the performance:** task completion time can be dramatically decreased if many robots cooperate to do the tasks in parallel.
- **Increasing reliability:** increasing the system reliability through redundancy because having only one robot may work as a bottleneck for the whole system especially in critical times. But when having multiple robots doing a task and one fails, others could still do the job.
- **Simplicity in design:** having small, simple robots will be easier and cheaper to implement than having only single powerful robot.

These benefits have attracted many researchers from academia and industry to investigate the applicability of MRS in many pertinent areas of industrial and commercial importance such as intelligent security [1], search and rescue [2], surveillance [3], humanitarian demining [4], environment monitoring [5, 6] and health care [7].

In order to develop and deploy robust MRS in real-world applications, a number of challenging problems needs to be solved. These problems include, but are not limited to, task allocation, group formation, cooperative object detection and tracking, communication relaying and self-organization to name just a few. The following section discusses in details the task allocation problem as one of the challenging problems of MRS.

MRTA problem is one of the most challenging problems of MRS especially when it comes to heterogeneous unreliable robots equipped with different types of sensors and actuators and are required to perform various tasks with different requirements and constraints in an optimal way. This problem can be seen as an optimal assignment problem where the objective is to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints.

In spite of the great number of MRTA algorithms reported in the literature, important aspects have, to date been given little attention. These aspects include but are not restricted to allocation of complex tasks, dynamic task allocation, heavily constrained task allocation and heterogeneous allocation.

The main objective of this chapter is to provide a comprehensive review on challenging aspects of MRTA problem, recent approaches to tackle this problem and the future directions. The remainder of the chapter is organized as follows: Sect. 2 describes MRTA problem as one of challenging problems of MRS. Section 3 highlights different MRTA schemes and planning followed by discussing different organizational paradigms that can be used in Sect. 4. Section 5 reviews two well-known MRTA approaches, namely, metaheuristic-based and market-based approaches. Finally conclusion and future directions are summarized in Sect. 6.

2 Multi-robot Task Allocation (MRTA) Problem

MRTA problem addresses the question of finding the task-to-robot assignments in order to achieve the overall system goals [8, 9]. This can be divided into two sub-problems. First, how a set of tasks is assigned to a set of robots. Second, how the behavior of the robot team is coordinated in order to achieve the cooperative tasks efficiently and reliably. Because the problem of task allocation is a dynamic decision problem that varies in time with phenomena including environmental changes, the problem should be solved iteratively over time [10]. Thus, the problem of task allocation becomes more complex to tackle. The requirements of the particular domain under consideration affect the features and complexity of multi-robot task allocation problems [11].

2.1 Problem Formulation

As illustrated in Fig. 1, MRTA can be formulated as an optimal assignment problem where the objective is to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints.

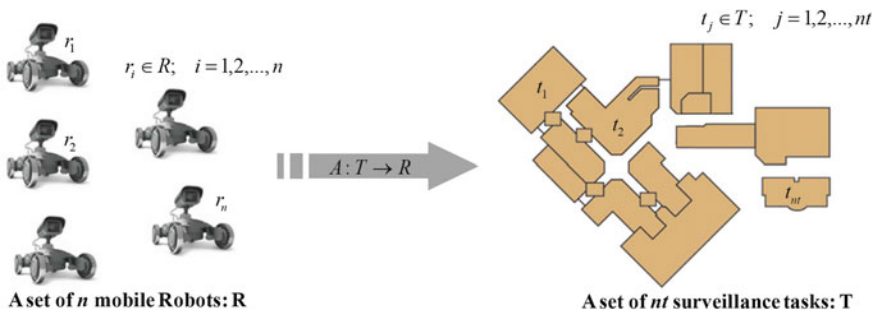


Fig. 1 MRTA problem

In this problem, it is given:

1. R : a team of mobile robots r_i ; $\{i = 1, 2, \dots, n\}$.
2. T : a set of tasks t_{ij} ; $\{j = 1, 2, \dots, nt\}$.
3. U : a set of robots' utilities, u_{ij} is the utility of robot i to execute task j .

For a single sensor task, the problem is to find the optimal allocation of robots to tasks, which will be a set of robot and task pairs [12]:

$$(r_1, t_1), (r_2, t_2), \dots, (r_k, t_k) \text{ for } 1 \leq k \leq m \quad (1)$$

For the general case, the problem is to find the optimal allocation of a set of tasks to a subset of robots, which will be responsible for accomplishing it [13]:

$$A : T \rightarrow R \quad (2)$$

In some MRTA approaches such as market-based approaches (Sect. 5.1), each robot $r \in R$ can express its ability to execute a task $t \in T$, or a bundle of tasks $G \subseteq T$ through bids $b_r(t)$ or $b_r(G)$. The cost of a bundle of tasks can be simply computed as the sum of costs of the individual tasks:

$$b_r(G) = \sum_{k=1}^f b_r(t_k) \{t_k \in G\} \quad (3)$$

where f is the number of tasks of the bundle G . The group's assignment determines the bundle $G \subseteq T$ of tasks that each robot $r \in R$ receives.

2.2 Problem Modeling

Many methods have been proposed in the literature to model the MRTA problem as described in the following sections.

2.2.1 Discrete Fair Division

The MRTA problem can be seen as an example of a Fair Division Problem [14]. Given a set of N robots (r_1, r_2, \dots, r_N) and a set of tasks S . It is required to divide S into N shares (s_1, s_2, \dots, s_N) so that each robot gets a fair share of S . A fair share is a share that, in the opinion of the robot receiving it, is worth $1/N$ of the total value of S .

Fair division problems can be classified depending on the nature of the set of shares S into two kinds:

- Indivisible tasks, such that each item should be given entirely to a single robot.
- Divisible tasks, which are often modeled as a subset of a real space. Additionally, the set to be divided may be homogeneous, or heterogeneous. Thus the set of the dividable tasks should be given to homogeneous or heterogeneous robot team.

Two different schemes have been reported in the literature to deal with discrete fair division problems. The first scheme is called the method of sealed bids [15] at which each bidder submits a secret sealed bid. The bids are kept private until the closing of the bidding period. After the auction closes, the bids are opened by the auctioneer and the auction winner is determined. The winner will be the one with the highest bid price. The second scheme for discrete fair division is the method of markers [16]. In this method, the tasks could be arranged in a linear fashion. This may be the case when a large number of small tasks need to be shared. The N available robots indicate their opinion as regard a fair division by placing $N - 1$ markers and agree to accept any segment of the tasks that lies between any pair of their consecutive markers. The next step is to find the leftmost among the consecutive markers. The owner of this marker receives the first segment (the one from the left end and up to the marker itself) and all the remaining markers of that robot are removed from further consideration. This step is repeated until all robots received what they think a fair share in their opinion. Fair division-based MRTA approach is described in [17]. This approach only addresses the allocation of a single global task between a group of heterogeneous robots.

2.2.2 Optimal Assignment Problem (OAP)

As mentioned previously, MRTA can be seen as an example of optimal assignment problem. In this type of problems [18], given a set of robots R , a set of tasks T the goal is to maximize the profit $W(rt)$ made by assigning robot r to task t . By adding virtual tasks or robots with zero profitability, it can be assumed that R and T have the same size n which can be written as $R = r_1, r_2, \dots, r_N$ and $T = t_1, t_2, \dots, t_N$.

Mathematically, the problem can be stated: given an $n \times n$ matrix W , find permutation π of $1, 2, 3, \dots, n$ for which:

$$\sum_{i=1}^n w(r_i t_{\pi(i)}) \text{ is maximized} \quad (4)$$

Such matching from R to T is called an optimal assignment. Related to multi-robot task allocation, the goal is to assign the set of robots R to the set of tasks T such that the profit is maximized [19].

2.2.3 ALLIANCE Efficiency Problem (AEP)

The alliance algorithm is a mono-objective optimization algorithm that was first used to solve NP problems [20]. It has been generalized to tackle any mono-objective

optimization problem [8], and has been used to solve artificial life problems and robotics problems [9]. In this algorithm, several tribes, with certain skills try to conquer an environment that offers resources needed for their survival [20]. Two features characterize each tribe: the skills and the resources necessary for survival.

A tribe t is a tuple (x_t, s_t, r_t, a_t) composed of:

- a point of solution space x_t
- a set of skills $s_t = [s_{t,1}, s_{t,2}, \dots, s_{t,N_s}]$ that depends on the values of N_s objective function $S = [S_1, S_2, \dots, S_{N_s}]$ evaluated at x_t :

$$s_{t,i} = S_i(X_t) \forall i = 1, 2, \dots, N_s \quad (5)$$

- a set of resource demands $r_t = [r_{t,1}, r_{t,2}, \dots, r_{t,N_R}]$ that depends on the values of the N_R . Generally there is only one constraint function:

$$r_t = R(x_t) \quad (6)$$

- An alliance a_t that records the IDs of the tribes allied to tribe t .

2.2.4 Multiple Traveling Salesman Problem

The Multiple Traveling Salesman Problem (mTSP) is a generalization of the Traveling Salesman Problem (TSP) in which more than one salesman is allowed [21, 22]. Given a set of cities, and m salesmen, the objective of is to determine a tour for each salesman such that, starting from the same base city, each salesman visits at least one city and returns to the base city so as to minimize the total cost. The cost could be distance or time. A comprehensive study of 32 formulations for the multiple traveling salesman problem is investigated in [22] considering their relative performances. The presented formulations differ by the way the sub-tour elimination constraints are modeled. Thus, the models can be accordingly classified as follows: (i) those that are based on the ranking of the cities; (ii) those that are based on explicit time-indexed variables for ranking the cities, and (iii) those that are based on multi commodity flow constructs.

The main difference in the mTSP is that instead of a single salesman, a number of salesmen m are given. The salesmen are required to cover all the available nodes and return back to their starting node such that each salesman make a round trip. The mTSP can be formally defined on a graph $G = (V, A)$ where V is the set of n nodes and A is the set of arcs. Let $C = (c_{ij})$ be the distance matrix associated with A . Assuming the more general case which is an asymmetric mTSP, thus $c_{ij} \neq c_{ji} \forall (i, j) \in A$. The mTSP can be formulated as follows [23]:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is used in the tour} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$\text{minimize } \sum_{i=1}^n \sum_{j=1}^n c_{ij} \times x_{ij} \quad (8)$$

$$\sum_{j=2}^n x_{1j} = m \quad (9)$$

$$\sum_{j=2}^n x_{j1} = m \quad (10)$$

$$\sum_{i=1}^n x_{ij} = 1, j = 2, \dots, n \quad (11)$$

$$\sum_{j=1}^n x_{ij} = 1, i = 2, \dots, n \quad (12)$$

$$x_{ij} \in \{0, 1\}, \forall (i, j) \in A \quad (13)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |\text{subTour}| - 1, \quad \forall S \subseteq V \setminus \{1\}, \text{subTour} \neq \phi \quad (14)$$

where (8) represents the objective function which is the summation of the total distance traveled, (9) and (10) ensures that exactly m salesmen departed their starting node and returned back. Equations (11)–(13) are the usual assignment constraints. Finally, (14) is the sub-tour elimination constraint.

A number of variations of the original mTSP were introduced by different researchers to accommodate the mTSP to their problems. These variations included the following [23]:

- **Salesmen starting node:** all the salesmen may start from a single depot node and then all of them must return back to the same node or every salesman can start from a certain node, and thus each salesman must return back to his starting node.
- **Number of salesmen:** the number of salesmen used in different applications varies according to the type and requirements of the application itself. In some applications, the number of salesmen is dynamic such that after each iteration the number of salesman may or may not change.
- **City time frame:** in some applications the task of the salesman is not only to visit the city, but also to stay in the city for a certain time in order to move to the next city.
- **Fair division of salesmen:** another variation of the general mTSP is the addition of constraints that specify the maximum number of cities or the maximum distance that can be traveled by a single salesman. This variation can be used in applications that are concerned with the fair division of the available resources (salesmen).

In [24], the MRTA problem is modeled as a multi-traveling salesman problem considering that robots play the roles of the salesmen and the tasks are the same as cities.

3 MRTA Schemes and Planning

As illustrated in Fig. 2, existing task allocation schemes can be categorized according to several dimensions [25]:

- Single task (ST) versus multi-task (MT), related to the parallel task performing capabilities of robots,
- Single robot (SR) versus multi-robot (MR), related to the number of robots required to perform a task, and
- Instantaneous assignment (IA) versus time extended assignment (TA), related to the planning performed by robots to allocate tasks.

ST means that each robot is capable of executing as most one task at a time, while MT means that some robots can execute multiple tasks simultaneously. Very similarly, SR means that each task requires exactly one robot to achieve it, while MR means that some tasks can require multiple robots. In IA approaches the available information concerning the robots, the tasks, and the environment permits only an instantaneous allocation of tasks to robots (i.e. tasks independence is a strong assumption). These approaches are sometimes used in order to avoid the need for highly computationally scheduling algorithms. At the other extreme, there are continuous task allocation or time extended assignment approaches where more information is available, such as the set of all tasks that will need to be assigned. Because robots have to reason about the dependencies between tasks, TA is more demanding from a planning perspective [26].

From the perspective of planning, there are two common approaches to the task allocation problem: decompose-then-allocate and allocate-then-decompose. In the first technique, the complex mission is decomposed to simple sub-tasks and then these sub-tasks are allocated to the team members based on their capability and availability to complete the sub-tasks as required [27, 28]. In this type of techniques, the cost of the final plan cannot be fully considered, because the task decomposition

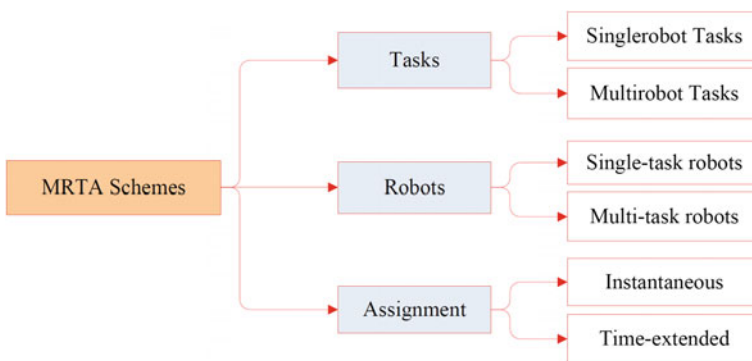


Fig. 2 MRTA schemes

is done without knowing to whom tasks will be allocated. Another disadvantage of this type is inflexibility to changes in the designed plan. So, the plan designed by the central agent cannot be rectified even if it is found costly. On the other side, in the allocate-then-decompose approach [28], the complex tasks are allocated to mobile sensors, and then each mobile sensor decomposes the awarded tasks locally. The main disadvantage of this approach is the allocation of all tasks to only one mobile sensor and thus, the preferred task decomposition is purely dependent on the plan of that mobile sensor, which increases the possibility of reaching a sub-optimal solution. It may be more beneficial to allocate tasks to more than one mobile sensor in order to consider different plans for the required task. While the decompose-then-allocate and the allocate-then-decompose methods may be capable of finding feasible plans, there are drawbacks to both approaches.

4 Organizational Paradigms

MRTA approaches can be classified according to team organizational paradigm. This paradigm shows how the multiple robots/agents of the system are organized by specifying the relationships and interactions among the agents and the specific roles of each agent within the system. The following subsections describe centralized and decentralized organizational paradigms.

4.1 Centralized Approaches

In this type of systems, each agent maintains a connection to one central agent that allocates the tasks to the other agents. Thus, the separate agents send all the information they have to this central agent, which in turn processes this information and sends the appropriate commands to these agents to execute the assigned tasks. The advantages of this type include the reduction of duplication of effort, resources, and increased savings of cost and time [29]. Although the centralized systems are widely implemented in the literature [30], there are many disadvantages that restrict the use of this paradigm in multi-robot task allocation. The lack of robustness is one of the most important disadvantages of the centralized system. In other words, if the central agent fails, the whole system will fail. Also, the system scalability is restricted because all the agents are connected to the central agent that is considered as a bottleneck. Practically, fully centralized approaches can be computationally intractable, brittle, and unresponsive to changes. Thus, for multi-robot task allocation problems where number of robots and tasks are small and the environment is static or global state information is easily available, centralized approaches are the best-suited solution. The centralized approach is one of the most widely reported approaches in the literature for solving the task allocation problems [31]. In [32], a centralized algorithm is proposed to solve the MRTA problem in order to assign tasks to mobile

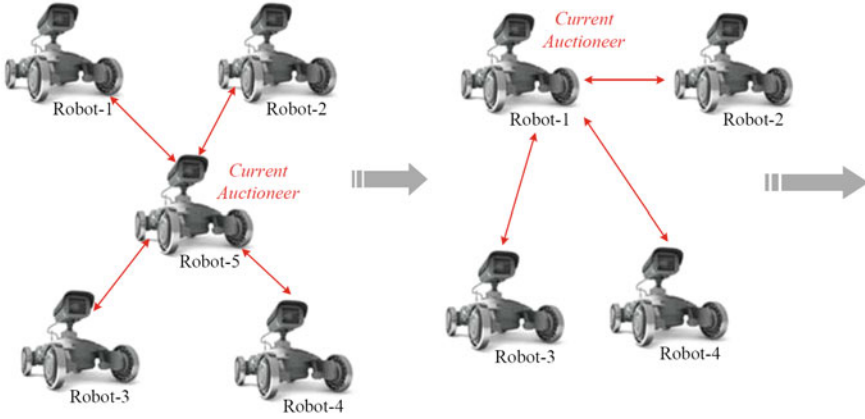


Fig. 3 Hierarchical organizational paradigm

robots to extend the life time of the sensor network. Also in [33], a centralized approach is introduced to solve the MRTA for the inspection problem in an industrial plant. Fair division-based MRTA approach described in [17] is another centralized algorithm that allocates a single global task between a group of heterogeneous robots.

4.2 Decentralized Approaches

Decentralization is the process of dispersing the administrative tasks and authorities between the agents of the multi-agent system [29]. In this type of configuration, there is no centralized agent that allocates the tasks to the other agents. Each agent is communicating its information with the other agents. Each agent can work on its own without major consideration of the other agents. Also, sometimes an agent of the decentralized system needs to exchange information with other agents in order to achieve its mission efficiently in harmony with other agents.

Many decentralized approaches are proposed to solve MRTA problem. In [34], the authors proposed a decentralized implementation of the Hungarian method proposed in order to solve the MRTA problem. In [35], two decentralized auction-based approaches, namely, the consensus-based auction algorithm and the consensus-based bundle algorithm are proposed for solving the MRTA problem of a fleet of autonomous mobile robots. Also an evolutionary computation decentralized approach is proposed for solving the MRTA problem using genetic algorithm in [36]. Hierarchical market-based approach has been proposed in [26] as a decentralized approach for MRTA problem. As shown in Fig. 3, the tasks are allocated initially to the robots 1; 2; 3; and 4 via a central auctioneer 5. Each robot can hold auctions in rounds for the tasks it won in the initial auction.

The main advantage of the decentralized system is its robustness. For example, in distributed systems, if one of the agents fails, the other agents are still working on their own and/or cooperatively with others [37]. As there is no centralized agent as a bottleneck, new agents can be added in case of failure for example. This means that scalability is no longer an issue in decentralized systems. In general, decentralized approaches have many advantages over centralized approaches such as flexibility, robustness, and low communication demands. However, because a good local solution may not sum to a good global solution, decentralized approaches can produce highly sub-optimal solutions.

5 MRTA Approaches

The following subsections describe two of most commonly used MRTA approaches, namely market-based approaches and optimization-based approaches.

5.1 Market-Based Approaches

Market-based approach gained a considerable attention within the robotics research community because of several desirable features, such as the efficiency in satisfying the objective function, robustness and scalability [9]. The market-based approach is an economically inspired approach that provides a way to coordinate the activities between robots/agents. It is mainly based on the concept of auctions. In economic theory, an auction is defined by any mechanism of trading rules for exchange [38]. An auction is a process of assigning a set of goods or services to a set of bidders according to their bids and the auction criteria. Auctions are common and simple ways of performing resource allocation in a multi-agent system.

Market-based approaches for MRTA problem involve explicit communications between robots about the required tasks. Robots bid for tasks based on their capabilities. The negotiation process is based on market theory, in which the team seeks to optimize an objective function based upon robots utilities for performing particular tasks [13]. The following subsections provide more details about auctions, and winner determination strategies.

5.1.1 Auctions

Auctions, in one form or another, have been used in societies throughout history to allocate scarce resources among individuals and groups. Generally, any protocol that allows agents to indicate their interest in one or more resources or tasks is considered an auction. This makes auctions very important to consider when tackling many applications. Moreover, auctions provide a general theoretical structure for understanding resource allocation among self-interested agents. Since auctions are simply mechanisms for allocating goods, there are various types of auction that can

achieve this goal. These auctions can be divided into two main categories, simple-good auctions and combinatorial auctions. Figure 4 shows taxonomy for different auctions types.

5.1.2 Auction Design

The auction has several designs that can be used to solve multi-robot task allocation problem. In this section some of these designs are defined and discussed.

- **Contract Net Protocol (CNP):** CNP is a task-sharing protocol in multi-agent systems. It specifies the interaction between agents for autonomous competitive negotiation through the use of contracts. Thus, CNP allows tasks to be distributed among multi-agents. Smith in 1980 was the first one to apply CNP to a simulated distributed acoustic sensor network [39]. The contract net protocol enables dynamic distribution of information via three methods:
 - Nodes can transmit a request directly to another node for the transfer of the required information.
 - Nodes can broadcast a task announcement in which the task is a transfer of information.
 - Nodes can note, in its bid on a task, that it requires particular information in order to execute the task.

The details of CNP algorithm as shown in Fig. 5 and it works as follows:

- **Announcement stage:** an agent takes up the role of the coordinator/auctioneer and announces the tasks or a set of tasks to be available for bidding.
- **Submission stage:** after calculating the individual utility values based on the objective function, individual agents/bidders communicate this value to the coordinator agent.

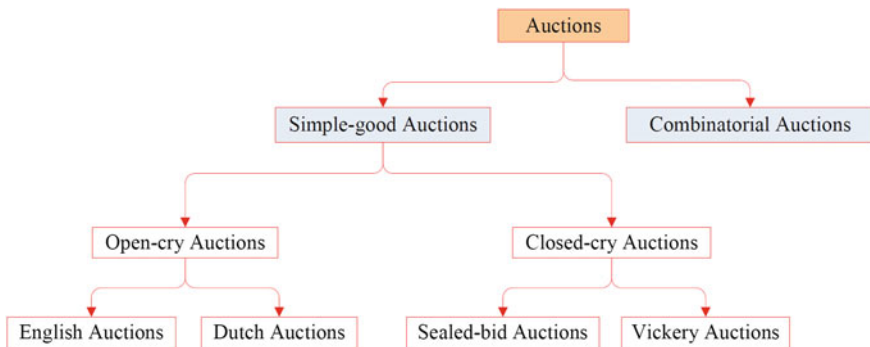


Fig. 4 Auctions types

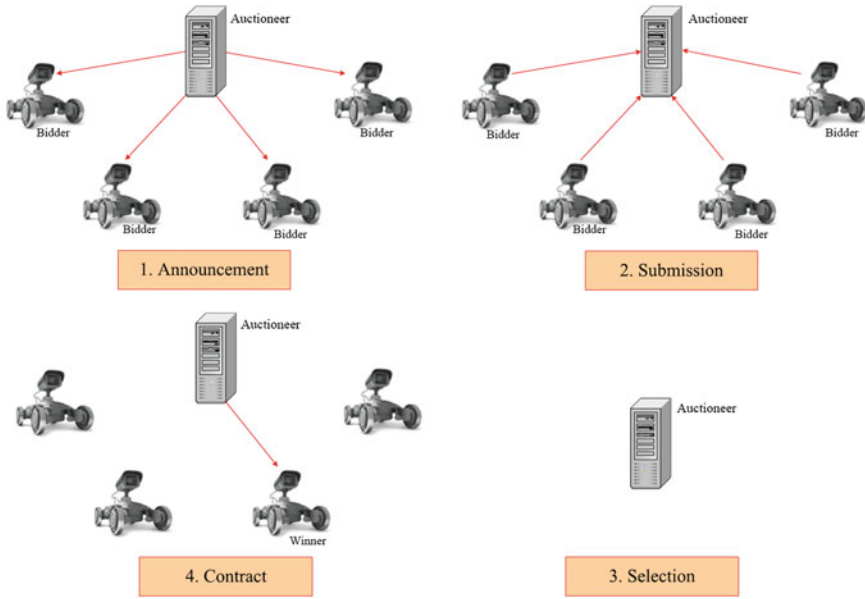


Fig. 5 Contract net protocol algorithm

- **Selection stage:** after receiving all the bids from the bidders, the job of the auctioneer is to evaluate the received bids based on an optimization strategy to determine the winning agent.
- **Contract stage:** the winning agent get assigned by a contract to execute the task and the process loops all over again.

The main contribution of the contract net protocol is that it offers structuring high-level interactions between nodes for cooperative task execution. Wherever, the main drawback is that each agent is a self-interested agent; meaning that the final solution may be the best for the agents involved, but not for the group as whole [40].

- **Trader-Bots:** Trader-Bots approach applies market economy techniques for generating efficient and robust multi-robot coordination in dynamic environments. The top level of Trader-Bots architecture consists of multiple traders; one trading agent for each robot, plus other trading agents representing operators or other resources such as computers and sensors. Each trader has the ability to reason about tasks and resources in order to make rational decisions when negotiating contracts [12]. The objective functions for this approach are designed to reflect the nature of the application domain. These functions reflect the domain characteristics in terms of priorities for task completion, hard deadlines for relevant tasks, and acceptable margins of error for different tasks. The goal in this algorithm is to have a team of robots which can complete the tasks efficiently maximizing overall profits, while maximizing the individual profits for each robot as well. The main advantages of

this algorithm are self-organization, learning and adaptation, and robustness [12]. The trader representing a robot is called a RoboTrader, and the trader representing an operator is called an OpTrader. For single-task contracts, Trader-Bots uses first-price sealed-bid auctions in generating the efficient coordination [13]. Trader-Bots makes use of two modes of contracts; subcontracts and transfers.

- **Subcontract:** the bidder is agreeing to perform a task for the seller at a given price, and must report back to the seller upon completion to receive payment.
- **Transfer:** the right to perform a task is sold for a price and the payment goes from the seller to the buyer upon the awarding of the contract.

5.1.3 Pros and Cons of Market-Based Approaches

Market-based approaches have several advantages such as [12, 38]:

- **Efficiency:** one of the greatest strengths of market-based approaches is their ability to utilize the local information and preferences of their participants to arrive at an efficient solution given limited resources [32]. Market-based approaches have elements that are centralized and other elements that are distributed [32]. Thus they can produce efficient solutions by capturing the respective strengths of both distributed and centralized approaches. It has been shown in [26, 32, 41, 42] that efficient solutions can be produced by market approaches with respect to a variety of team objective functions.
- **Robustness:** as mentioned previously, fully centralized approaches employ a single agent to coordinate the entire team in a multi-agent system. They may suffer from a single point of failure, and have high communication demands. Market-based approaches implemented based on decentralized paradigm do not require a permanent central coordinator agent and therefore there is no common-mode failure point or vulnerability in the system. These approaches can be made robust to several types of malfunctions, including complete or partial failures of agents [32, 38, 43].
- **Scalability:** as mentioned before, the computational and communication requirements of market-based approaches are usually manageable, and do not prohibit these systems from providing efficient solutions because they are not fully centralized systems. Thus, as the size of the inputs in the system increases, these approaches can still provide an efficient solution [32]. Market-based approaches can scale well in applications where the team mission can be decomposed into tasks that can be independently carried out by small sub-teams [38]. However and as concluded in [42], optimization-based approach outperforms market-based approach in handling large-scale MRTA scenario (fifty tasks and fifteen robots).
- **Online input:** market-based approaches are able to seamlessly incorporate the introduction of new tasks [41]. Market-based approaches can often incorporate online tasks by auctioning new tasks as they are introduced to the system or generated by the agents themselves [38].

- **Uncertainty:** market-based systems are able to operate in unknown and dynamic environments by allowing team members to adapt cost estimates over time, and reallocate tasks when appropriate [44].

Although market-based approaches have many advantages, they are not without their disadvantages. Perhaps the biggest drawback of market-based approaches is the lack of formalization in designing appropriate cost and revenue functions to capture design requirements [45]. Also, negotiation protocols, developing appropriate cost functions, and introducing relevant penalty schemes can complicate the design of the market approach [12]. In domains where fully centralized approaches are feasible, market-based approaches can be more complex to implement, and can produce poorer solutions [45]. Also, when fully distributed approaches suffice, market-approaches can be unnecessarily complex in design and can require excessive communication and computation [45].

5.2 *Optimization-Based Approaches*

Optimization is the branch of applied mathematics focusing on solving a certain problem in the aim of finding the optimum solution for this problem out of a set of available solutions. This set of available solutions is restricted by a set of constraints, and the optimum solution is chosen within these constrained solutions according to a certain criteria. This criteria defines the objective function of the problem that quantitatively describes the goal of the system [46]. There is a wide variety of optimization approaches available, and the use of these approaches depends on the nature and the degree of complexity of the problem to be optimized. Moreover the optimization-based approaches algorithms have higher potential for exploring new search areas in the search space because the randomness of the algorithm variables which also enable an enhanced performance when dealing with noisy input data [47–49]. Figure 6 shows a general classification of optimization techniques [50].

Deterministic techniques follow a rigorous procedure and its path and values of both design variables and the functions are repeatable. For the same starting point, they will follow the same path whether you run the program today or tomorrow. Deterministic techniques include numerical and classical methods such as graphical methods, gradient and hessian based methods, derivative-free approaches, quadratic programming, sequential quadratic programming, penalty methods, etc. They also include graph-based methods such as blind/uninformed search and informed search methods.

Stochastic techniques always have some randomness. These techniques can be classified into trajectory-based and population-based algorithms. A trajectory-based metaheuristic algorithm such as simulated annealing uses a single agent or solution which moves through the design space or search space in a piece-wise style. A better move or solution is always accepted, while a not-so-good move can be accepted with certain probability. The steps or moves trace a trajectory in the search space,

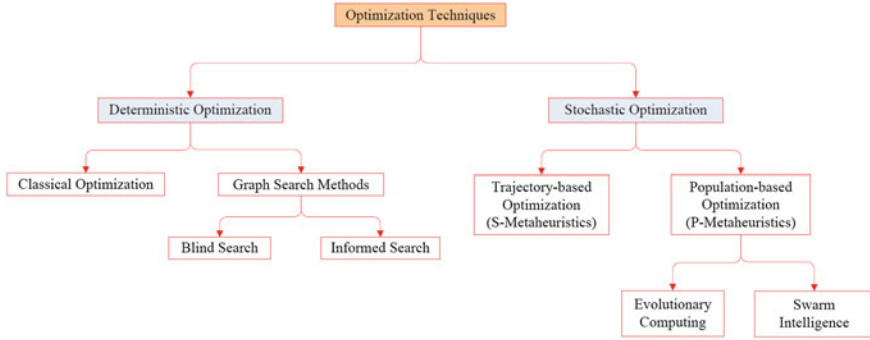


Fig. 6 Optimization techniques

with a non-zero probability so that this trajectory can reach the global optimum. On the other hand, population-based algorithms such as genetic algorithms, ant colony optimization and particle swarm optimization use multiple agents to search for an optimal or near-optimal solution.

By reviewing the literature, it was found that different optimization approaches have been used in order to solve the general task allocation problems and MRTA problem. In [51], a mixed integer linear programming optimization approach was used in order to allocate heterogeneous robots for maximizing the coverage area of the regions of interest. Also in [52], a mixed integer linear programming approach was used for solving the task allocation problem in the context of UAV cooperation. In [53, 54], a simulated annealing approach was used to solve the allocation of multi-robot system through formulating the MRTA problem as mTSP. In [55, 56], simulated annealing incorporated with other heuristic approaches was used to allocate a set of tasks to a number of processors in computer system problems.

Different optimization approaches were also used for solving the task allocation problem. For example, population-based approaches such as the genetic algorithm was used in [57] for providing a feasible solution for a group tracking system which is capable of tracking several targets rather than individual targets. Genetic algorithm was also used in [58] to provide a solution for the time extended task allocation of multi-robots in a simulated disaster scenario. Ant colony optimization, another technique of the population-based optimization approaches, was used in [59] to solve the task allocation problem of MRS. In [60], ant algorithm was used in the context of multi-robot cooperation for the aim of solving the task allocation problem.

The task allocation problem was also solved using hybrid optimization approaches such as tabu search with random search method in [56] and tabu search with noising method in [61]. In [62], a simultaneous approach for solving the path planning and task allocation problems for a MRS is proposed, where simulated annealing and ant colony optimization approaches were investigated and applied for solving the problem.

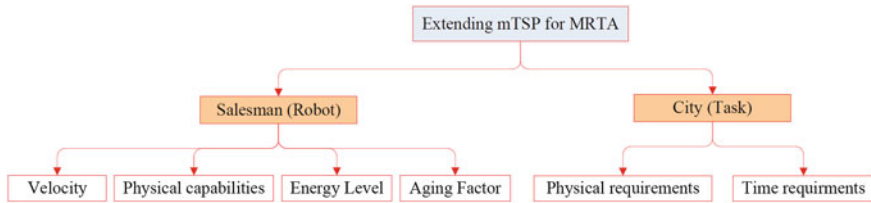


Fig. 7 Extending mTSP formulation for MRTA

Trajectory-based metaheuristics and population-based metaheuristics have been proposed in [42]. These two optimization-based approaches were extensively tested over a number of test scenarios showing the efficacy of the proposed algorithms in handling complex heavily constrained MRS applications that include extended number of heterogeneous tasks and robots. Figure 7 illustrates the extension of the mTSP formulation to accommodate the requirements of the MRTA problem. Since most real MRS applications require heterogeneous robots of different capabilities, it was a must to consider the heterogeneity of the robots in the proposed approach. Four main features of the robot were considered and thus were added to the traveling salesman in the implementation phase. The four features are velocity of the robot; robot capabilities; energy level of the robot and aging factor (efficiency). In the same manner, the mTSP formulation for solving the MRTA problem needed to be adapted to handle the heterogeneity of the tasks and therefore it was a must to add extra features to the cities. The added features to the cities are task requirements and minimum time required to finish the task.

A comparative study between metaheuristics-based and market-based approaches is reported in [42]. This study quantitatively evaluates the performance of these two approaches in terms of their ability to produce feasible solutions that maximize overall system performance and decrease the costs and the ability to handle real-world constraints such as time constraints and robot capabilities-task requirements matching constraints. Scalability is also considered as an evaluation metric in this study. The experimental results using different scenarios show that metaheuristics approaches outperform market-based approach in the scalability scenario while both approaches provide nearly similar results in the constraints handling scenarios. The results of this comparative study is presented in Table 1. The suitability of the algorithms depends on the required application domain of the MRTA problem. The stars evaluate the algorithm's efficiency in handling the application scenario, i.e. more stars means better algorithm [42].

6 Conclusion

This chapter reviewed the different challenging aspects of multi-robot task allocation problem, the recent approaches to tackle this problem and the future directions. The

Table 1 MRTA approaches applicability results

Scenario/algorithm	Market-based	Simulated annealing	Genetic algorithm
Small-scale	★	★	★
Medium-scale	★	★★★	★★
Large-scale	★	★★★	★★
Capabilities matching	★★	★	★
Time matching	★	★	★
Heavily constraints	★	★	NA

chapter also discussed two well-known approaches, metaheuristic-based and market-based approaches that are used extensively to solve the MRTA problem. Many of the reviewed approaches are capable of handling complex task allocation with different forms of constraints such as time constraints and robot capabilities-task requirements matching constraints.

Multi-robot task allocation with ability to handle more complex constraints is still open and needs to be tackled by researchers. These complex constraints can be categorized into environment-related constraints, robot-related constraints and task-related constraints. Environment-related constraints include, but are not limited to, the dynamic and unpredictable nature of the environment and its partial observability and complexity. Robot-related constraints can include limited sensing/acting range, limited radio coverage and partial malfunctions. Task-related constraints may include time extended tasks and tight tasks that cannot be decomposed into single robot tasks or tasks with precedence constraints.

References

1. Yi-Lin, L., Kuo-Lan, S.: Multi-robot-based intelligent security system. *Artif. Life Robot.* **16**, 137–141 (2011)
2. Nagatani, K., Okada, Y., Tokunaga, N., Kiribayashi, S., Yoshida, K., Ohno, K., Koyanagi, E., et al.: Multi-robot exploration for search and rescue missions. In: *IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, pp. 373–387 (2009)
3. 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.* **71**, 423–444 (2013)
4. Khamis, A., ElGindy, A.: Minefield mapping using cooperative multirobot systems. *J. Robot.* **2012** (2012)
5. Espina, M.V., Grech, R., De Jager, D., Remagnino, P., Iocchi, L., Marchetti, L., King, C., et al.: Multi-robot teams for environmental monitoring. *Innovations Defence Support Syst.* **336**, 183–209 (2011)
6. Shkurti, F., Xu, A., Meghjani, M., Gamboa Higuera, J.C., Girdhar, Y., Giguere, P., Dudek, G., et al.: Multi-domain monitoring of marine environments using a heterogeneous robot team. *Intell. Robots Syst. (IROS)* 1747–1753 (2012)

7. Shiomi, M., Kamei, K., Kondo, T., Miyashita, T., Hagita, N.: Robotic service coordination for elderly people and caregivers with ubiquitous network robot platform. In: IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO) (2013)
8. Lerman, K., Jones, C., Galstyan, A., Mataric, M.: Analysis of dynamic task allocation in multi-robot systems. *Int. J. Robot. Res.* **25**, 225–241 (2006)
9. Tang, F., Parker, L.E.: A complete methodology for generating multi-robot task solutions using asymptotic and market-based task allocation. In: International Conference on Robotics and Automation, IEEE, pp. 3351–3358 (2007)
10. Gerkey, B., Mataric, M.: A framework for studying multi-robot task allocation (2003)
11. Korsah, G.A., Stentz, A., Dias, M.B.: A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robot. Res.* **32**, 1495–1513 (2013)
12. Dias, M.B.: Trader-bots: a new paradigm for robust and efficient multirobot coordination in dynamic environments. Ph.D. Thesis, Robotics Institute, Carnegie Mellon University (2004)
13. Zlot, R., Stentz, A.: Market-based multirobot coordination for complex tasks. *Int. J. Robot. Res.* 73–101 (2006)
14. de Longueville, M.: A Course in Topological Combinatorics. Springer, New York (2012)
15. Vatsolaki, P., Tsalpatouros, A.: Ewos: A sealed-bid auction system design and implementation for electricity interconnector capacity allocation. In: Information, Intelligence, Systems and Applications (IISA) (2013)
16. Kim, D., So, Y., Kim, S.: Study of marker array list method for augmented reality service based smart home. *Int. J. Smart Home* **5**, 51–64 (2011)
17. Higuera, J., Dudek, G.: Fair subdivision of multi-robot tasks. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 3014–3019 (2013)
18. Kuhn, H.W.: The hungarian method for the assignment problem. *Nav. Res. Logistics Quart.* **2**, 83–97 (1955)
19. Nam, C., Shell, D.: Assignment algorithms for modeling resource contention and interference in multi-robot task-allocation. In: Presentation at IEEE International Conference on Robotics and Automation (ICRA) (2014)
20. Lattarulo, V., Parks, G.T.: A preliminary study of a new multi-objective optimization algorithm. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2012)
21. Xu, Z., Wen, Q.: Approximation hardness of min-max tree covers. *Oper. Res. Lett.* **38**, 169–173 (2010)
22. Sarin, S.C., Sherali, H.D., Bhootra, A.: New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Oper. Res. Lett.* **33**, 62–70 (2005)
23. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. In: *Omega*, vol. 34, pp. 209–219. Elsevier (2006)
24. Hussein, A., Khamis, A.: Market-based approach to multi-robot task allocation. In: International Conference on Individual and Collective Behaviors in Robotics (ICBR), IEEE (2013)
25. Dasgupta, P.: Multi-robot task allocation for performing cooperative foraging tasks in an initially unknown environment. *Innovations Defence Support Syst.*, **338**, 5–20 (2011)
26. Khamis, A.M., Elmogy, A.M., Karray, F.O.: Complex task allocation in mobile surveillance systems. *J. Intell. Robot. Syst.* **64**, 33–55 (2011)
27. Aylett, R., Barnes, D.: A multi-robot architecture for planetary rovers. In: Proceedings of the 5th ESA Workshop on Advanced Space Technologies for Robotics and Automation, pp. 1–3 (1998)
28. Botelho, S.C., Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: International Conference on Robotics and Automation, pp. 1234–1239 (1999)
29. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.* **19**, 281–316 (2004)
30. Brumitt, B.L., Stentz, A.: Grammps: a generalized mission planner for multiple mobile robots in unstructured environments. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1564–1571 (1998)

31. Al-Yafi, K., Lee, H., Mansouri, A.: Mtap-masim: a multi-agent simulator for the mobile task allocation problem. In: IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, pp. 25–27 (2009)
32. Coltin, B., Veloso, M.: Mobile robot task allocation in hybrid wireless sensor networks. In: Intelligent Robots and Systems (IROS), pp. 2932–2937 (2010)
33. Liu, C., Kroll, A.: A centralized multi-robot task allocation for industrial plant inspection by using a* and genetic algorithms. *Artif. Intell. Soft Comput.* 466–474 (2012)
34. Giordani, S., Lujak, M., Martinelli, F.: A distributed algorithm for the multi-robot task allocation problem. In: Trends Applied Intelligent System, Springer, pp. 721–730 (2010)
35. Han-Lim, C., Brunet, L., How, J.: Consensus-based decentralized auctions for robust task allocation In: IEEE Transactions on Robotics, pp. 912–926 (2009)
36. Ping-an, G., Zi-xing, C., Ling-li, Y.: Evolutionary computation approach to decentralized multi-robot task allocation. In: International Conference on Natural Computation (ICNC), pp. 415–419 (2009)
37. Elmogy, A.: Market-based framework for mobile surveillance systems. Ph.D. Thesis, University of Waterloo (2010)
38. Zlot, R.M.: An auction-based approach to complex task allocation for multirobot teams. Ph.D. Thesis, Carnegie Mellon University (2006)
39. Smith, R.: Communication and control in problem solver. In: IEEE Transactions on Computers (1980)
40. Alibhai, Z.: What is contract net interaction protocol? IRMS Laboratory, SFU (2003)
41. Dias, M., Stentz, A.: Opportunistic optimization for market-based multirobot control. In: International Conference on Intelligent Robots and Systems, pp. 2714–2720 (2002)
42. Badreldin, M., Hussein, A., Khamis, A.: A comparative study between optimization and market-based approaches to multi-robot task allocation. *Adv. Artif. Intell.* **2013**, 1–11 (2013)
43. Dias, M.B., Stentz, A.: A free market architecture for distributed control of a multirobot system. In: International Conference on Intelligent Autonomous Systems, pp. 115–122 (2000)
44. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: International Conference on Robotics and Automation (ICRA) (2002)
45. Dias, M., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. In: Proceedings of the IEEE, pp. 1257–1270 (2006)
46. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization: Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht (2000)
47. Spall, J.C.: Stochastic optimization. In: Handbook of Computational Statistics, Springer, New York, pp. 173–201 (2012)
48. Diwekar, U.: Optimization under uncertainty. In: Introduction to Applied Optimization. Springer, New York, pp. 1–54 (2008)
49. Lenagh, W.H.: Multi-robot task allocation: a spatial queuing approach. Ph.D. dissertation, University of Nebraska, Omaha (2013)
50. Khamis, A.: Ece457a: Cooperative and Adaptive Algorithms. University of Waterloo. Springer, Canada (2014)
51. Atay, N., Bayazit, B.: Mixed-integer linear programming solution to multi-robot task allocation problem. Washington Univ. St. Louis, Tech. Rep. (2006)
52. Darrah, M., Niland, W., Stolarik, B.M.: Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission. In: American Institute of Aeronautics and Astronautics, pp. 2324–2334 (2005)
53. Mosteo, A.R., Montano, L.: Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions. In: Workshop on Network Robot Systems Toward Intelligent Robotic Systems Integrated with Environments (2006)
54. Mosteo, A.R.: Multi-robot task allocation for service robotics: from unlimited to limited communication range. Ph.D. Thesis, Universidad de Zaragoza (2010)
55. Juedes, D., Drews, F., Welch, L., Fleeman, D.: Heuristic resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems. In: Parallel and Distributed Processing Symposium, pp. 1631–1638 (2004)

56. Kmiecik, W., Wojcikowski, M., Koszalka, L., Kasprzak, A.: Task allocation in mesh connected processors with local search meta-heuristic algorithms. In: *Intelligent Information and Database Systems*, Springer, pp. 215–224 (2010)
57. Shea, P.J., d Alexander, K., Peterson, J.: Group tracking using genetic algorithms. In: *Proceedings of the International Society Information Fusion (2003)*
58. Jones, E.G., Dias, M., Stentz, A.: Time-extended multi-robot coordination for domains with intra-path constraints. *Auton. Robots* **59**, 41–56 (2011)
59. Wang, J., Gu, Y., Li, X.: Multi-robot task allocation based on ant colony algorithm. *J. Comput.* **7**, 2160–2167 (2012)
60. Ding, Y., He, Y., Jiang, J.: Multi-robot cooperation method based on the ant algorithm. In: *IEEE Swarm Intelligence Symposium*, pp. 14–18 (2003)
61. Chen, W., Lin, C.: A hybrid heuristic to solve a task allocation problem. *Comput. Oper. Res.* **27**, 287–303 (2000)
62. Liu, D.K., Kulatunga, A.K.: Simultaneous planning and scheduling for multi-autonomous vehicles. In: *Evolutionary Scheduling*, Springer, pp. 437–464 (2007)

Efficient Trajectory Planning for WSN Data Collection with Multiple UAVs

D. Alejo, J.A. Cobano, G. Heredia, J. Ramiro Martínez-de Dios and A. Ollero

Abstract This chapter discusses the problem of trajectory planning for WSN (Wireless Sensor Network) data retrieving deployed in remote areas with a cooperative system of UAVs (Unmanned Aerial Vehicles). Three different path planners are presented in order to autonomously guide the UAVs during the mission. The missions are given by a set of waypoints which define WSN collection zones and each UAV should pass through them to collect the data while avoiding passing over forbidden areas and collisions between UAVs. The proposed UAV trajectory planners are based on Genetics Algorithm (GA), RRT (Rapidly-exploring Random Trees) and RRT* (Optimal Rapidly-exploring Random Trees). Simulations and experiments have been carried out in the airfield of Utrera (Seville, Spain). These results are compared in order to measure the performance of the proposed planners.

1 Introduction

Research and development of Unmanned Aerial Vehicles (UAVs) and aerial robots have been increasing in the last years due to the advantages that UAVs present over ground vehicles in terms of maneuverability and accessibility to remote areas. Different kinds of missions with UAVs have been addressed such as: surveillance [1], structure assembly [2], fire detection and monitoring [3], data collection [4], etc.

D. Alejo (✉) · J.A. Cobano · G. Heredia · J.R. Martínez-de Dios · A. Ollero
University of Seville, Camino de Los Descubrimientos S/N, Seville, Spain
e-mail: dalejo@us.es

J.A. Cobano
e-mail: jcobano@us.es

G. Heredia
e-mail: guiller@us.es

J.R. Martínez-de Dios
e-mail: jdedios@us.es

A. Ollero
e-mail: aollero@us.es

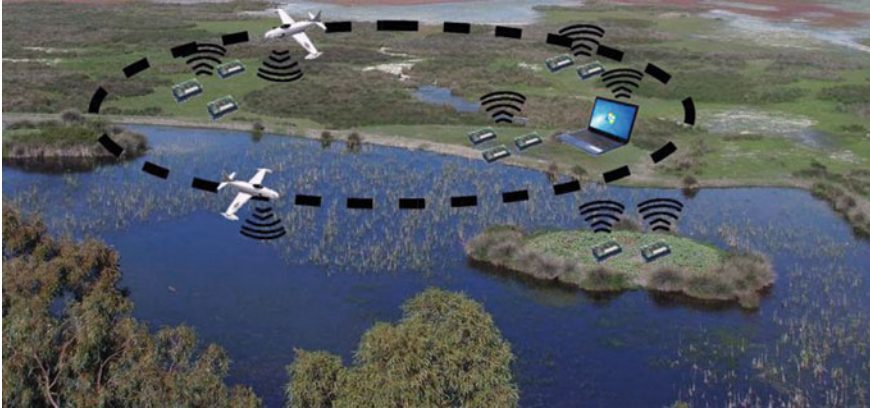


Fig. 1 Environmental monitoring in the PLANET project. WSN distributed over a large area have to be visited by a group of UAVs for data collection purposes

The work presented in this chapter has been developed within the framework of the UE-funded PLANET project (<http://www.planet-ict.eu>). The main objective of PLANET is the design, development and validation of an integrated platform to enable the deployment, operation and maintenance of large-scale/complex systems of heterogeneous networked Cooperating Objects, including Wireless Sensor and Actuator Networks and mobile robots. The platform support optimal and adaptive deployment and operation by means of mobile cooperating objects, i.e. vehicles, networked with static nodes. The platform has been validated in the monitoring of the Doñana Biological Reserve with very high ecological value and very sensitive to the impact of pollution.

The motivation of this work is the environmental monitoring in areas with difficult accessibility (see Fig. 1). A scenario in which a high number of nodes that have been deployed at known locations is considered. Particularly, this chapter addresses the planning of collision-free trajectories for data collection with UAVs in these zones. Firstly, the deployed nodes are grouped into groups taking into account the location of the nodes, their transmission ranges and message rates [4]. Each group has its WSN collection zone. The centers of the collection zones are considered as UAV trajectory waypoints. Then, a trajectory planning algorithm is used to ensure that the UAVs pass through the WSN collection zones taking into account the UAV kinematic constraints while avoiding collisions with among UAVs.

Therefore, trajectory planning algorithms should adapt to possible changes in the mission. Moreover, the algorithms should also be computationally efficient in order to ensure a suitable solution in a given time. Four different planners have been implemented to address this problem.

The main difficulty that has to be considered in order to develop the planners derives from the fact that the problem of trajectory planning is NP-hard [5, 6]. In addition, differential constraints given by the model of the UAV should be considered to make the problem tractable. Sampling-based techniques, as opposed to combina-

torial planning, are usually preferred in these NP-hard problems. These planning schemes are suitable when the solution space is hard to model or unknown a priori because of its dynamic nature. Furthermore, planning optimal collision-free trajectories for UAVs leads to optimization problems with multiple local minima in most cases and, thus, local optimization methods as gradient-based techniques are not well suited to solve it. The application of sampling-based techniques is an efficient and effective alternative for this problem.

The first of the proposed path planners is based on genetic algorithms (GA). These algorithms are randomized algorithms that can give quasi-optimal solutions and that can be adapted to multiple different problems, such as pattern recognition [7], control system design [8], shortest path routing [9] and collision avoidance [10]. They are a particular kind of evolutionary techniques that randomly generate and evolve a population of individuals. The goodness of the individuals is calculated by means of a criteria function.

The second path planner is based on Rapidly-exploring Random Tree (RRT) planning algorithm [11]. RRT planning algorithms have moderate computational requirements and can be easily adapted to changes in the environment conditions. Therefore, these algorithms are used to quickly compute a collision-free solution. The drawback of this method is that the generated path is not optimized.

It would be desirable to refine the solution obtained with RRT algorithm in the available computational time. The third path planner, based on RRT* (Optimal Rapidly-exploring Random Trees) planning algorithm [12], is designed to fulfill this purpose. The main advantage of RRT* is that it computes smoother trajectories than the ones obtained with RRT, it can optimize the length of the paths and the generated paths are more predictable.

The main contributions of this chapter are summarized in the following points:

- The chapter proposes four UAV trajectory planning methods for WSN data collection. The proposed methods adopt sampling-based approaches in contrast to traditional schemes that rely on heuristic-based reasoning. Besides, three of the four proposed methods are classified into the class named “anytime techniques”, i.e. methods that ensure a valid solution in a very short time and uses the remaining time to refine the solution searching for a better solution.
- The chapter compares the four UAV trajectory planning methods and evaluate them in massive simulation experiments. The RRT_i^* planner shows the best performance and therefore it was selected for experimentation in real hardware experiments.
- The chapter validates the RRT_i^* method in real experiments performed with a Megastar fixed-wing UAV. The validation shows that the distance between UAVs and center of the WSN collection zones are lower than the collection zone radius, enabling correct data retrieval.

The chapter is organized into seven sections. The state of the art is presented in Sect. 2. The background of the implemented planners is given in Sect. 3. The

addressed problem is described in Sect. 4. Section 5 presents the three path planners implemented. The simulations and experiments performed are shown in Sects. 6 and 7, respectively. Finally, the conclusions are detailed in Sect. 8.

2 State of the Art

This section briefly summarizes the state of the art in the main topics involved in the chapter: collection of WSN data using UAVs and UAV path planning.

Data collection using UAS has been an attractive research topic due to its wide potentialities. Some works have proposed theoretical and/or simulated analysis and architectures, see e.g. [13], which describes an architecture for the integration of WSNs and aerial vehicles or [14].

In the simplest approach, the deployed nodes gather and buffer the readings. When the UAS flies near the nodes it sends a beacon message and the nodes send the readings in reply. The UAS collects the data. Experiments with the mentioned approach are described in [15, 16]. In [4] the scalability of the basic scheme is increased grouping the deployed nodes. The groups and their collection zones are pre-computed taking into account the nodes locations and radio coverage, among others. This approach has been extended for multi-UAV cooperation purposes in this paper.

Work [17] proposes a UAS-WSN cooperation scheme where the results of the WSN operation are used to update the UAS flight plan and, at the same time, the UAS trajectory is considered in the operation of the WSN in order to improve the data collection performance. These UAS-WSN cooperation strategies require advanced UAS planning methods. In contrast, the work presented in this paper focus on the safe trajectory planning for multi-UAV systems.

UAV path planning with safety requirements has been largely investigated. A* and Theta* algorithms are efficient classic alternatives, but they are not suitable for multi-UAV planning and require a discretization of the state space that could be prohibitive in open 3D spaces.

This paper focus on multi-UAV planning, in particular in solving a Conflict Detection and Resolution (CDR) problem. Considering this problem with multiple mobile UAVs is NP-hard [18]. Sampling-based techniques match particularly well when the solution space is hard to model or unknown a priori because of its dynamic nature.

A large number of algorithms have been developed to avoid applying the brute force approach. In [19, 20] a detailed survey on CDR techniques and planning algorithms is presented, respectively. CDR methods can be coarsely classified in deterministic and stochastic.

Deterministic CDR methods perform an exploration of the feasible solution set using search procedures usually directed by the local behavior of the optimization function. These methods include non-linear programming (NLP) [21], integer programming [22, 23], dynamic programming if the problem can be broken down into simpler sub-problems [24], collocation methods reducing the number of dimensions

of the problem [25, 26], among many others. These algorithms do not adapt well to changes of the environment.

Stochastic CDR methods perform a random exploration of the problem by using random variables [27]. This involves random objective functions or random constraints. These methods include random search methods [28], evolutionary computation methods [29, 30], particle swarm optimization [31], ant colony optimization methods [32], simulated annealing [33], among others. Rapidly-exploring Random Tree (RRT) planning algorithm is also commonly used for path planning [12, 34]. RRT is constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree. This approach is suitable for path planning problems that involve obstacles and differential constraints.

3 Background

This paper is focused on path planning. It aims to apply state of the art planning methods to real systems and make replanning procedures online. In this section, the basic applied algorithms are explained.

The planification problem consists in searching for a path between an initial configuration of one or more vehicles q_{init} to a desired configuration q_{goal} while avoiding collisions with static obstacles and between vehicles. Let C be the set of possible configurations of the system, C_{free} be the set of configurations that are collision-free, while $C_{obs} = C \setminus C_{free}$ is the set of colliding configurations.

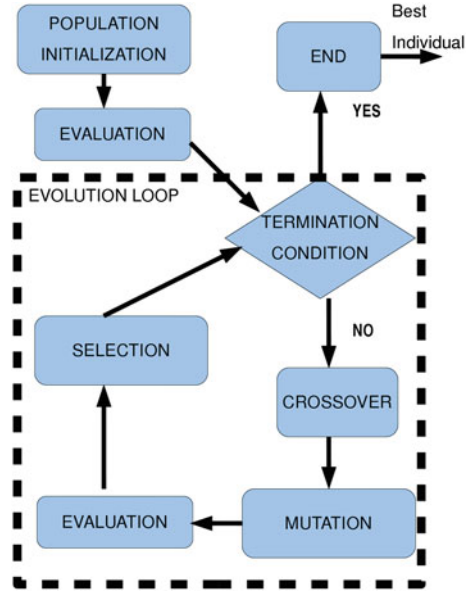
In this paper, GA optimization method has been applied in order to solve the path planning problem, as well as three randomized sampled-based planners: RRT, RRT and RRT_i^* .

3.1 Genetic Algorithm

Genetic algorithm (GA) is a heuristic search that mimics the process of natural selection, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

Figure 2 represents the basic flowchart of the GA. First, a population of candidate solutions to an optimization problem (called individuals or phenotypes) is created and then evolves toward better solutions. This evolution starts from an initial population generated randomly and is an iterative process. The population in each iteration is a generation. The fitness of every individual in the population is evaluated in each generation. The fitness is defined by the value of the objective function in the optimization problem being solved. Moreover, each individual has a set of properties which can be randomly mutated and recombined (crossover) to generate a new individuals, which are usually called offspring. Then, the offspring and old

Fig. 2 Flowchart of the planning algorithm based on genetic algorithms



population compete in order to generate a new generation with the same size. This new generation will be used in the next iteration of the algorithm.

Commonly, the algorithm terminates when either a maximum number of generations has been produced, a satisfactory fitness level has been reached for the population, or a maximum allowed computation time is reached.

3.2 RRT

RRT is a planning algorithm first proposed in [34]. The basic RRT algorithm is shown in Algorithm 1. Note that some procedures are necessary for the algorithm to be run. Below you can find the list of procedures.

- **Nearest**(G, q). Searches for the closest vertex in the graph G to the configuration q .
- **Steer**(q_1, q_2). Obtains the configuration q_3 that is the closest to q_2 integrating the model from q_1 one step.
- **CollisionFree**(q_1, q_2). Returns **true** if the path that unites q_1 and q_2 is collision free.
- $q_{rand} = \mathbf{SampleFree}()$. Returns a configuration $q_{rand} \in C_{free}$.

RRT starts a tree by creating the root in the starting configuration (q_{init}) and extends the tree by generating random samples (x_{rand}) of the configuration space and by making the tree extend to that new point. When the new sample is generated,

the closest node to it is selected and the tree is extended from this sample and a new node is added (x_{new}). This new node is generated by integrating the model proposed in [35] from v_{near} with a random control signal. If the path between v_{near} and q_{new} is collision-free this node is added to the tree. This procedure is repeated until the new node is sufficiently near from the final state q_{goal} . Note that this algorithm ensures that the generated paths are flyable because they are generated by integrating the UAV model.

Many different variants of the RRT algorithm have been proposed over the years, in particular the variants that propose the growth of two trees, one starting from the goal point and one from the starting point claim to outperform basic RRT [36]. These variants are called bi-RRT. Another common improvement is to make a bias in the sampling procedure towards the goal, i.e. taking the goal as the sampled state with a configured probability (usually 10%).

Algorithm 1 Basic RRT algorithm

Require: $RRT(q_{init}, q_s)$

```

1:  $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset$ 
2: repeat
3:    $x_{rand} \leftarrow SampleFree()$ 
4:    $v_{nearest} \leftarrow Nearest(G = (V, E), x_{rand})$ 
5:    $x_{new} \leftarrow Steer(v_{nearest}, x_{new})$ 
6:   if  $CollisionFree(v_{nearest}, x_{new})$  then
7:     // Add the new vertex and the connection
8:      $V \leftarrow V \cup \{x_{new}\}$ 
9:      $E \leftarrow E \cup \{(v_{nearest}, x_{new})\}$ 
10:  end if
11: until  $q_s \in G = \{V, E\}$ 
12: return  $G = \{V, E\}$ 

```

3.3 RRT*

The main drawback of the RRT algorithm, when applied to mobile robots, is that the basic RRT yielded to randomized like motions that were not properly optimized and difficult to forecast. In order to overcome these drawbacks, RRT* planning algorithm makes two main modifications to the original algorithm [12] as shown in Algorithm 2.

First, when a new sample is generated, the algorithm attempts to connect it not only to the nearest neighbor but also to a set of neighbors that are close enough. Only the connection that minimizes the length of the path between the new sample and the starting configuration is added to the tree (steps 9–16).

The other modification is called the *rewiring step*. In this phase, the current cost of the neighbors of the new sample is compared to the cost that would be obtained

by traveling through the new sample. If this new cost is lower than the current cost, the graph is rewired (steps 20–23).

Some extra functions are necessary for RRT* algorithm to work. These are:

- **Cost** ($n \in V$). Associates the node n with its calculated cost.
- **c(Path)**. Gives a cost to a calculated path. In the basic version the cost is the distance of the path.
- **Near** (G, q, d). Returns a set of vertices $N = \{n \in V \mid \text{dist}(n, q) < d\}$.

Algorithm 2 RRT* algorithm

Require: RRT(q_{init}, q_s)

```

1:  $G = \{V, E\}$ 
2:  $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset$ 
3: repeat
4:    $x_{rand} \leftarrow \text{SampleFree}()$ 
5:    $v_{nearest} \leftarrow \text{Nearest}(G, x_{rand})$ 
6:    $x_{new} \leftarrow \text{Steer}(v_{nearest}, x_{new})$ 
7:   if  $\text{CollisionFree}(v_{nearest}, x_{new})$  then
8:      $V \leftarrow V \cup \{x_{new}\}$ 
9:     // Connect along a minimum-cost path
10:     $U \leftarrow \text{Near}(G, x_{new}, \eta)$ 
11:     $v_{min} \leftarrow v_{nearest}; c_{min} \leftarrow \text{Cost}(v_{nearest}) + c(\text{Path}(v_{nearest}, x_{new}))$ ;
12:    for all  $u \in U$  do
13:      if  $\text{CollisionFree}(u, x_{new})$  and  $\text{Cost}(u) + c(\text{Path}(u, x_{new})) < c_{min}$  then
14:         $v_{min} \leftarrow u; c_{min} \leftarrow \text{Cost}(u) + c(\text{Path}(v_{nearest}, x_{new}))$ 
15:      end if
16:    end for
17:     $E \leftarrow E \cup \{(v_{min}, x_{new})\}$ 
18:    // Rewire vertices
19:    for all  $u \in U$  do
20:      if  $\text{CollisionFree}(x_{new}, u)$  and  $\text{Cost}(x_{new}) + c(\text{Path}(x_{new}, u)) < \text{Cost}(u)$  then
21:         $v_{parent} \leftarrow \text{Parent}(u)$ 
22:         $E \leftarrow (E \setminus \{(v_{parent}, u)\}) \cup \{(x_{new}, u)\}$ 
23:      end if
24:    end for
25:  end if
26: until  $q_s \in G$ 
27: return  $G$ 

```

4 Description of the Problem

The scenarios considered in PLANET project consist of several UAVs in a common airspace. The scenario contains forbidden flight zones, for example due to presence of protected species. These forbidden flight zones will be modelled as static obstacles that the UAV trajectories must avoid. The rest of UAVs in the same common airspace

are considered as mobile obstacles. A mission is considered safe if during the flight, the separation between each UAV and obstacles is greater than a given safety distance (minimum separation). If a collision is detected, the trajectory of the UAVs should be updated to avoid the collision.

Therefore, the goal is to compute collision-free trajectories while minimizing the changes of the trajectory of each UAV involved in WSN data collection. Assume that the data collection zones corresponding to the groups assigned to one UAV are denoted as $\{W_1, W_N\}$, so a UAV trajectory is defined by this sequence of waypoints. The proposed method computes safe trajectories for UAVs passing through $\{W_1, W_N\}$ in presence of static and mobile obstacles. The solution only considers the addition of intermediate waypoints. The trajectory computed should ensure that the UAV passes through the WSN collection zones to a minimum distance from the center of the zone.

Although the trajectories are planned to be free of collisions, UAVs can deviate from these trajectories due to wind or other disturbances. Thus, a collision detection and avoidance method is needed to check that the actual trajectory is free of collisions. A collision detection module should check if the actual UAV trajectory is free of collisions. If a collision with a static obstacle or with another UAV is detected, a new collision-free trajectory should be computed.

The information that the system needs in order to solve the problem is the following:

- Initial trajectory of each UAV
- Parameters of the model of each UAV
- Initial location and goal location of each UAV
- Look-ahead time to know the available computation time.

5 Collision-Free Trajectory Planning Algorithm

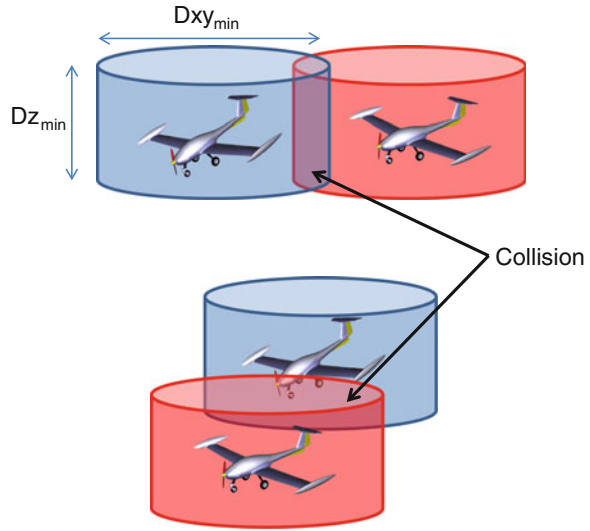
This section describes the blocks of the proposed algorithm to plan collision-free trajectories. First, a detection algorithm is implemented to detect possible collisions. Then, a collision-free trajectory planning algorithms based on genetic algorithms (GA), RRT and RRT* are implemented to solve the detected conflicts.

Each UAV is supposed to be surrounded by a cylinder which cannot be entered by the other UAVs or obstacles (see Fig. 3). A collision is detected if the horizontal separation between UAVs, the Euclidean distance in the XY plane, is lower than the $D_{xy_{min}}$ and at the same time the vertical separation is lower than $D_{z_{min}}$. This technique presents as advantages the low execution time and the need for few parameters to describe the system.

The resolution block is executed when an alert takes place. The alerts are produced for two main reasons:

1. Detection algorithm: a collision of one UAV with a static or mobile obstacle is detected.

Fig. 3 Detection algorithm: each aerial vehicle is described by a *cylinder*



2. Cooperation WSN-UAV: there is a change in the WSN collection zones and so the UAV flight plan changes.

In both cases the trajectory needs to be updated. This update is carried out by the collision-free trajectory planning algorithms based on GA [30], RRT [34] and RRT* [12]. The planning algorithm adds intermediate waypoints between WSN data collection zones in order to compute collision-free trajectories for each UAV by considering the kinematics constraints. Three planners have been implemented. Each planner is described in detail in next sections.

5.1 Genetic Algorithm

In this chapter, GA has been applied to path planning. Therefore, an individual will represent possible trajectories of all the UAV in the system. In this case, the individuals are coded by sequences of waypoints that represent a possible trajectory for each UAV.

Firstly, an initial population is randomly generated with an uniform distribution in the search space. It is important to point out that the initial location and the goal location are always the same for each trajectory of the UAV. Figure 4 (left) represents an example of 2D flight plan that could be generated in the initial population. The UAV starting location is $(0, 0)$ and the goal location is $(5, 0)$. The genome is given by the vector $V = (1, 1, 3.5, 2)$ that yields to two intermediate waypoints: $WP1 = (1, 1)$ and $WP2 = (3.5, 2)$.

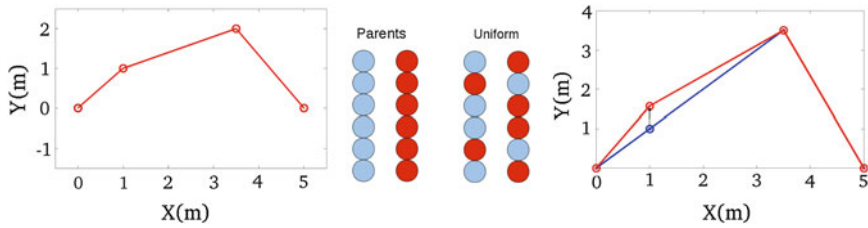


Fig. 4 Left Example of flight plan given by genome $G = (1, 1, 3.5, 2)$. Middle Crossover operator that has been used in this paper. Right Mutation of the second gene of the genome G (blue line) yields to a new genome $G_{mutation} = (1, 1.57, 3.5, 2)$ (red line)

Once the initial population has been generated, the following step is to evaluate the fitness of each candidate trajectory (individual). The proposed cost function that will define the fitness value to each candidate trajectory is proportional to the length of each trajectory. In addition a penalty is added if the trajectories yield to collision. So, the proposed cost function is:

$$J_i = L_i + P_{i,collision}, \quad (1)$$

where i indicates the i th iteration, L_i is the sum of the length of each UAV trajectory and $P_{i,collision}$ is the penalty added when a collision is detected.

In our implementation, the uniform crossover operator has been used. This operator randomly selects each gene from one parent as seen in Fig. 4 (middle). The mutation operator considered in the implemented algorithm randomly changes the mutated gene with a normal distribution centered in the original gene value and with configurable standard deviation. Figure 4 (right) represents an example of modification of the second gene of genome G (in blue), the new genome $G_{mutation}$ is represented in red.

By iteration of the selection and reproduction processes, the genetic algorithm ends up computing a near-optimal trajectory that ensures a collision-free trajectory, provided it exists.

5.2 RRT and RRT*

Planning algorithms to compute collision-free trajectories based on RRT and RRT* planning algorithms have been also implemented. They can be considered as a planning algorithm based on two steps:

1. RRT computes efficiently a first collision-free trajectory that solves the problem. This solution ensures safety but might have low performance.

2. RRT* is run to refine the solution from step 1 trying to improve its performance. This method generates a time-dependent improved trajectory while safety is maintained.

Firstly, a collision-free trajectory is generated by using an efficient algorithm based on RRT planning algorithm that ensures reactive fast timely reaction. RRT are greedy-based purely randomized techniques that can efficient find a collision-free trajectory.

Once a safe solution has been computed, the solution is refined trying to improve its performance (time-dependent improved trajectory). The concept of time-dependent improved trajectory means that other solution will be computed during the available time to improve the performance. This stage exploits this time to find a better trajectory. Therefore, the algorithm provides collision-free solutions even in case of requiring very short responses. In case of having more time the solutions are improved as the computation time increases. A RRT* planning algorithm is used to refine the initial solution. RRT* will improve the solution in the available horizon time. These algorithms provide a relatively good solution in short times and allow a simple implementation. These methods have been shown to have a potential to solve large-scale problems efficiently in a way that is not possible for deterministic algorithms.

However, RRT* planning algorithm uses a simple interpolation step in the steering procedure (see Sect. 3). Because of this, the new connections are checked by considering the kinematics constraints: curvature and maximum climb or descent rate. The connection attempts that do not fulfill this are discarded. The following kinematics constraints are considered from the parameters of the Megastar UAV:

$$R > 25 \text{ m}, \quad (2)$$

$$-5 \text{ m/s} < C < 5 \text{ m/s}, \quad (3)$$

where R is the turning radius and C is the climb or descent rate.

Last but not least, some improvements of the original versions of RRT and RRT* have been introduced in order to reduce the computational time of the planning algorithm and to generate paths with better clarification. In first stages of the algorithm, we propose the use of a non-uniform random distribution in order to explore first the zones that are near the WSN collection zones. In this case, a multivariate normal distribution has been used to produce the new samples. By using this sampling distribution, the explored space by the tree is much more oriented to the interesting areas. In addition, we bias the sampling towards the goals in the first stages of the algorithm.

Finally, some improvements proposed in [37] can be applied when a solution has been found. Firstly, the *localbias* when using the RRT* algorithm is used. The main idea is to sample in the surroundings of a random point of the solution path in order to encourage rewiring steps of the RRT* algorithm. Also, the *node rejection* technique has been implemented. In this case, a node is rejected if the sum of its

cost and the distance to the goal node is greater than the cost of the current solution. This technique is inspired in the A* algorithm [38]. The algorithm with the proposed additions is called RRT_i^* in order to distinguish it from the basic RRT* algorithm.

6 Simulations

Several simulations have been performed to validate the proposed algorithms. The planning algorithm based on genetic algorithm has been implemented in Matlab and C++ language and compiled with gcc-4.4.1. RRT and RRT* planning algorithms have been implemented in C++ by extending the Open Motion Planning Library (OMPL [39]) with the aforementioned improvements.

Taking into account the characteristics of the aerial vehicles involved in the simulations, the following dimensions of each cylinder are considered: $D_{xymin} = 50$ m and $D_{zmin} = 20$ m.

The main objective of this section is to compare the proposed planners and justify the election of the final planner.

6.1 GA Versus RRT

First simulation considers a UAV which should pass through four WSN collection zones (see Fig. 5). The generated trajectory with RRT is represented with a blue line. Note that the RRT planning algorithm generates several intermediate waypoints in order to ensure the correct data collection.

RRT planning algorithm is compared to genetic algorithm, both implemented in Matlab. Twenty simulations have been performed considering one UAV, four WSN groups and different obstacles. RRT ensures a fast initial solution and presents less computational load, 3.73 ± 0.21 s, than genetic algorithms, 6.89 ± 0.40 s. Moreover, RRT could add several intermediate waypoints to ensure that UAV passes through each WSN collection zone without collision. Genetic algorithms add only one intermediate waypoint between two consecutive waypoints. The computational time increases as more intermediate waypoints are added by Genetic algorithms.

6.2 RRT Results

Another scenario is considered with two UAVs (see Fig. 6). One collision is detected and RRT planning algorithm computes intermediate waypoints (green circles) to avoid it.

Another scenario with four WSN collection zones (WSN1, WSN2, WSN3 and WSN4) and two obstacles (red circle) is considered (see Fig. 7). UAV should pass

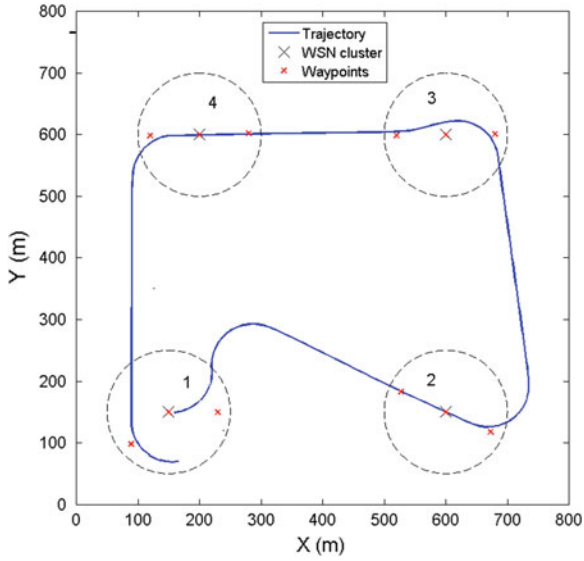


Fig. 5 Trajectory computed by RRT planning algorithm by considering four WSN collection zones

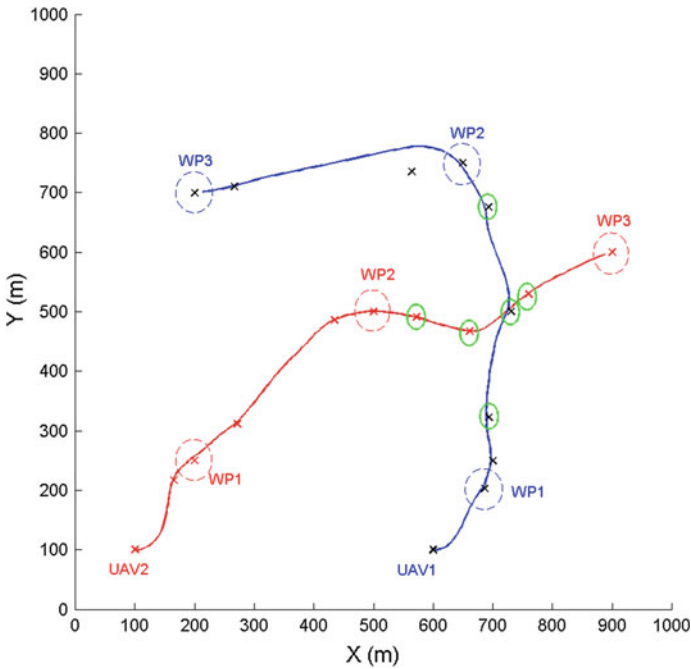


Fig. 6 Scenario with two UAVs. UAV1 (blue lines) should pass through the WP1, WP2 and WP3 (black) and UAV2 (grey lines) should pass through the WP1, WP2 and WP3 (red)

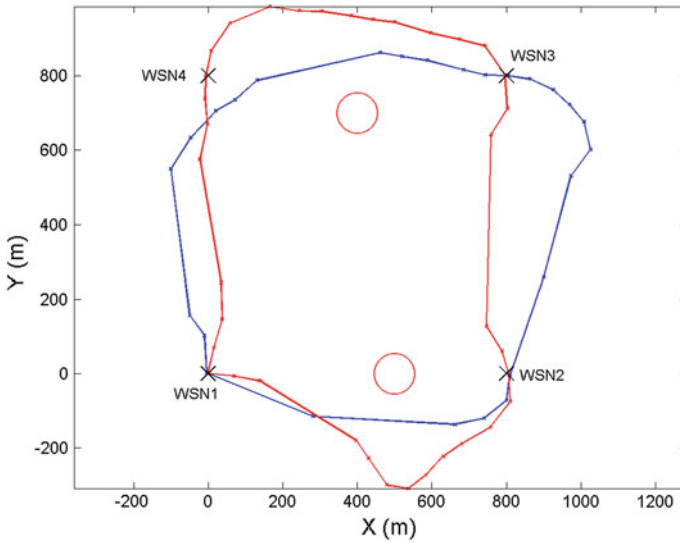


Fig. 7 Collision-free trajectories computed by RRT considering four WSNs and two obstacles

through the WSN collection zone considering a maximum distance. This distance considered is 50 m. Two different collision-free trajectories are shown in Fig. 7. Each trajectory passes through all the WSN groups. Note that the paths can change significantly, even though they are obtained with the same method.

The tree of the RRT planning algorithm is generated randomly, so every time the algorithm is executed a different collision-free trajectory is obtained from these trees. In addition, RRT lacks of an optimization method. In the original RRT algorithm the first obtained solution is not improved by any means, the remaining time is wasted in further exploring the search space.

6.3 RRT* Results and RRT Comparison

RRT* algorithm is able to improve the quality of the solution by rewiring the tree once the solution is found, overcoming the main issue of RRT algorithm. Thanks to this technique, the quality of the solution improves with the available time of execution. Figure 8 shows four different solutions considering two WSNs and five obstacles (red circle). The initial flight plan is defined by WSN1 and WSN2. Each solution is computed by considering different times of execution. The first solution (clear blue line) is computed in 5 s; the second one (blue line) in 10 s; the third one (red line) in 20 s; and the fourth one (green line) in 30 s. The improvement of the solution can be observed as the time of execution increases.

A comparison between RRT and RRT* planning algorithm has been performed. Figure 9 shows the solution trajectories computed by each one. Note that the trajectory

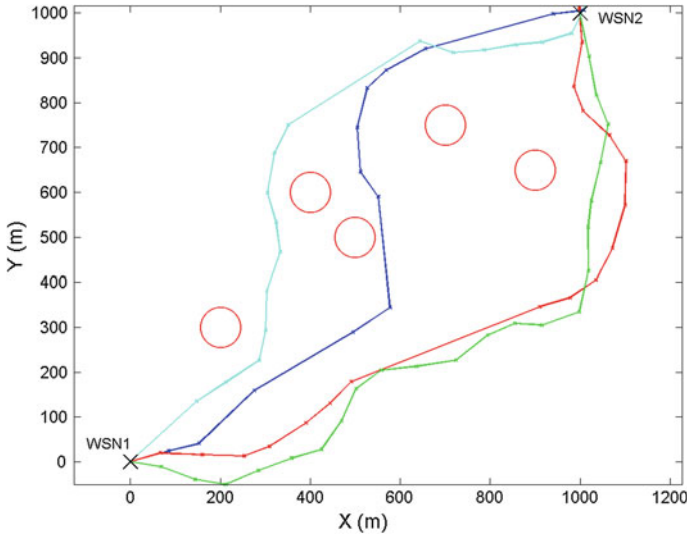


Fig. 8 Collision-free trajectories computed by RRT* considering different time of execution

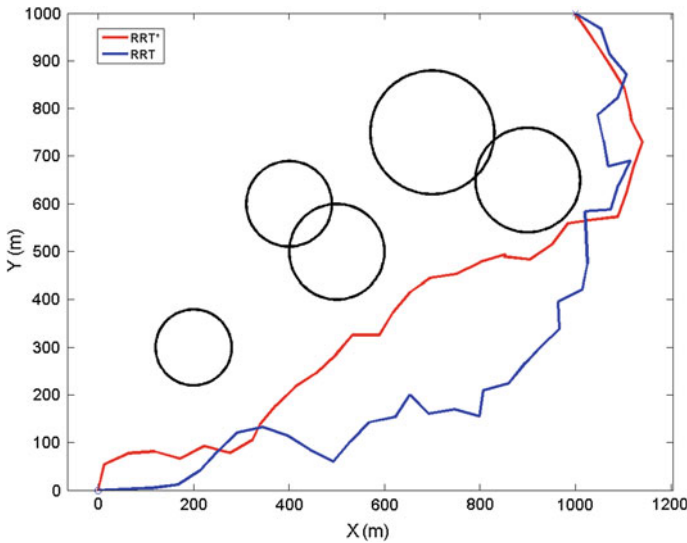


Fig. 9 Comparison between RRT (blue line) and RRT* (red line) generated trajectories

computed by RRT* planning algorithm is better and smoother than the computed by RRT.

The last simulation scenario is a multi-UAV scenario where two UAVs (UAV1 and UAV2) must fly over two collection nodes WSN1 and WSN2 respectively (see Fig. 10). This scenario has been also solved with RRT, RRT* and RRT_i^* algorithms.

Fig. 10 Last simulation scenario with WSN1 and WSN2 to be visited by UAV1 and UAV2 respectively. Comparison between RRT (blue) and RRT_i^* (red) generated trajectories. Static obstacles are represented with black circles. The minimum distance between UAVs in RRT_i^* is represented

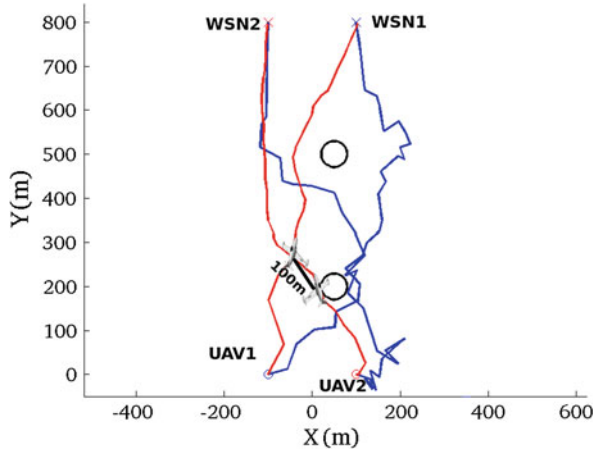


Figure 10 also represents the trajectory obtained with the RRT method in blue line and with RRT_i^* in red line. The minimum distance between UAV1 and UAV2 is represented and is greater than the safety distance ($D_{min_{xy}} = 50\text{ m}$).

Note that case the RRT_i^* is the final proposed planner. It starts with uniform sampling until a solution is found. When this happens, it automatically changes to local sampling. That is, sampling in the surroundings of a random point of the solution path with a Gaussian distribution with $\sigma = 5\text{ m}$. Also, the node rejection technique (see Sect. 5.2) is active.

Table 1 represents the mean and standard deviation obtained by the three methods in generating the first solution and the cost of the best solution after 30 s of execution. Each method is applied twenty times. It is noticeable that RRT_i^* outperforms both RRT* and RRT algorithms by a great margin when comparing the cost of the best obtained solution. However, RRT is the method that generates a faster first solution, while RRT* and RRT_i^* have similar performance.

Lastly, we compare RRT* and RRT_i^* in terms of optimization of the first solution. In fact, the improvement (final cost minus initial cost) obtained with RRT* has mean 10.0 and standard deviation 12.9. In contrast, the improvement of RRT_i^* has mean 113.2 and standard deviation 48.4. Note that RRT* does not improve the solution significantly, while RRT_i^* makes a better job. In addition, no significant improvement is done by RRT* when $t > 10\text{ s}$.

Table 1 Comparison of the execution time of the first solution (t) and cost of the best (c) solution when applying RRT, RRT* and RRT_i^*

Method	$ t $	σ_t	$ c $	σ_c
RRT	1.25	0.42	2301.3	175.1
RRT*	5.14	2.05	1931.4	138.6
RRT_i^*	4.88	2.37	1770	59.7

6.4 Simulation Remarks

As conclusions, this section shows that the RRT_i^* planner is the one that gives smoother and shorter trajectories when compared to the proposed RRT^* , RRT and GA methods. However, RRT planner is able to plan in the control space of the model, so flyable trajectories are ensured. And additionally, RRT spends less computational time when searching for the first solution. Thus, a good combination of both methods is to use the proposed RRT planner to find a first solution and then refine this solution by using the proposed RRT_i^* method.

7 Experiments

An experiment has been performed by using Megastar UAV (see Fig. 11), at the airfield of Utrera (Sevilla, Spain), in order to test and demonstrate the correct operation of the proposed planning algorithm in a real environment.

7.1 Hardware Setup

Figure 11 shows the Megastar and the Piper fixed-wing UAVs that were used in the experiments as well as the WSN Nodes deployed on the ground and on-board the UAVs. Both fixed-wing UAVs are propelled with gasoline motors and both of them are equipped with an embedded computer PC104 that executes the UAS controller. This controller communicates with the Control Station through a Radio Modem link at 433 MHz, where all the processing is carried out through. The Radio Modem B connects to the ground station via USB. This connection offers the localization



Fig. 11 *Left* Megastar fixed-wing UAV used in the experiments. *Right* Piper fixed-wing UAV used in experiments. Detail of the on-board and deployed WSN

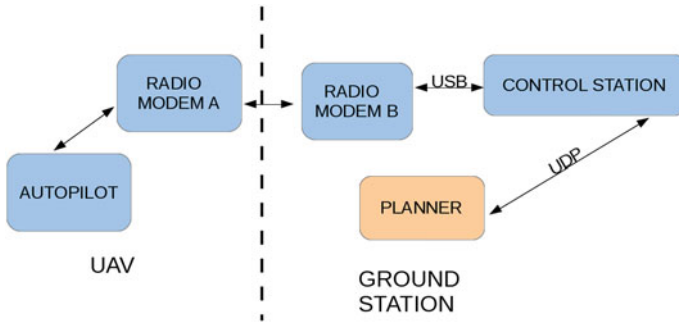


Fig. 12 *Left* Radio modem of the ground station. *Right* Communication diagram of the system

and status information of the UAV. Path planning is computed on the ground on an auxiliary computer that is linked to the Control Station using UDP protocol. The Control Station receives the list of waypoints computed by the path planning method and transmits them as the flight plan to the on-board controller. The on-board controller uses the list of waypoints as reference for the UAV controller. Figure 12 shows a scheme which represents the communications between the modules used in the experiments.

Motes from the *Mica2* family from *Crossbow Inc.* have been used. These systems have been selected because of requirements in size, weight and energy consumption. These motes use a 916 MHz transceiver at a rate of 9.6 Mbps. Their microprocessor is a *ATMega128* 8-bit @ 7 MHz. *Mica* family has different sensor boards such as the MTS400 with accelerometers and temperature and light sensors; or the MTS420 with GPS. However, in this experiment we are not interested in the real data provided by the WSN, but rather in establishing communications between deployed and on-board nodes.

7.2 Results

Figure 13 shows the location where the experiments have been carried out. Four different WSN collection zones were used $\{WP1, WP4, WP7, WP10\}$ and 5 WSN nodes were deployed inside each of them. Therefore, UAV should pass through the collection zones centered at $\{WP1, WP4, WP7, WP10\}$, see Fig. 13.

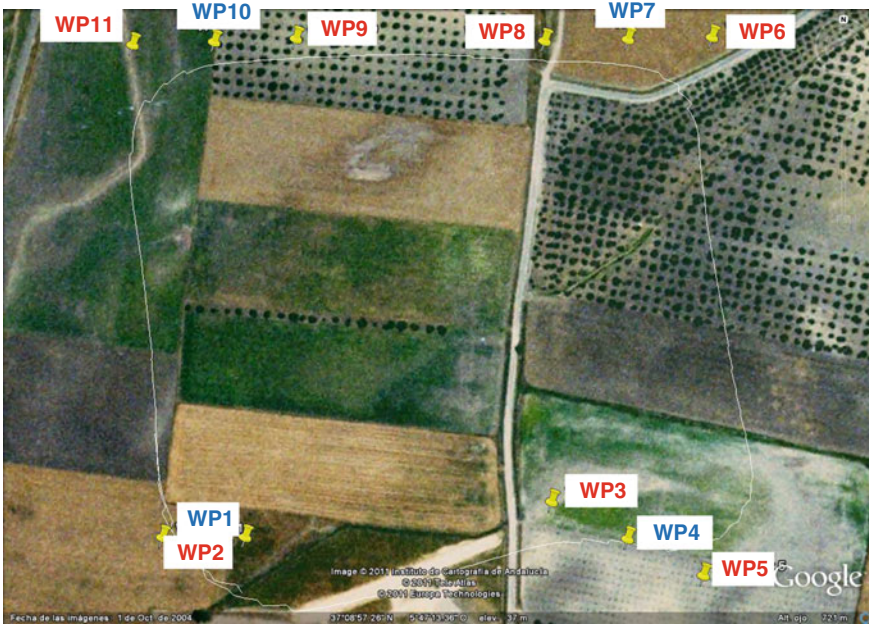


Fig. 13 Experiment performed at the airfield of Utrera (Seville, Spain). UAV should pass through WSN groups: WP1, WP4, WP7 and WP10

The RRT* method was executed and as result it provided a list of intermediate waypoints that forced that the UAV trajectory actually passed {WP1, WP4, WP7, WP10}. The list of intermediate waypoints were: WP2, WP3, WP5, WP6, WP8, WP9 and WP11, see Fig. 13. The complete list of waypoints was then used to generate the UAV flight plan, which was received by the UAV on-board controller. The UAV executed the flight plan and Table 2 shows the resulting distance between the UAV trajectory and the center of each WSN collection zone: the UAS passed through all collection zones at a distance suitable to perform WSN data collection. These distances obtained in all the UAV passes were similar and in all of them the UAS passed through all collection zones at a distance suitable for WSN data collection.

Table 2 Distance of pass of the UAV trajectory through each WSN collection zone

WSN collection zones	WP1	WP4	WP7	WP10
Distance (m)	13.00	6.04	13.28	11.23

8 Conclusions

Several collision-free trajectory planners based on a genetic algorithm, RRT and RRT* planning algorithm to perform applications of WSN (Wireless Sensor Network) data collection with Unmanned Aerial Vehicles have been presented. These kind of stochastic methods are good candidates in these applications because they are efficient computationally and can be easily adapted to changes in the environmental conditions. These algorithms compute trajectories between WSN collection zones to ensure that the UAV passes through the WSN collection zones.

This chapter shows that RRT and RRT* planning algorithm are more suitable in these scenarios. They quickly compute a collision-free solution and then the solution can be refined by RRT* considering the remaining time. RRT* computes smoother trajectories than the ones obtained with RRT planning algorithm. Also RRT_i^* algorithm has been presented that includes several improvements from the original RRT* algorithm.

Several simulations have been performed to check the validity of the proposed algorithms. In particular, several scenarios have been solved by using the path planning algorithm based on RRT* and RRT_i^* . The execution time is in the order of seconds demonstrating the efficiency of the proposed algorithm. RRT_i^* and RRT* algorithms are compared and the results show that RRT_i^* greatly outperforms RRT* in terms of the optimization of the initial solution.

Last but not least, a field experiment has also been done with the Megastar UAV in Utrera airfield to validate the approach in a real environment. The results of the experiment show that the UAV were able to collect data of four WSN nodes since the distance to the WSN nodes was lower than the maximum allowed deviation.

Future work will include performing a scalability analysis when applied to WSN deployed in a sparse area and with a higher number of UAVs in the system. Also, the use of parallelized planning algorithms such as *C-Forest* [40] seems convenient in order to take advantage of modern multi-core processors. To conclude, path pruning techniques such as the proposed in [41] can be used in a post-processing step to the paths obtained with RRT* method in order to get rid of unnecessary waypoints. However, this has to be studied carefully in multi-UAV applications, because it can produce unexpected collisions.

Acknowledgments This work was supported by the European Commission FP7 ICT Programme under the Project PLANET (European Commission FP7-257649-ICT-2009-5) and the RANCOM Project (P11-TIC-7066) funded by the Junta de Andalucía (Spain). David Alejo is granted with a FPU Spanish fellowship from the Ministerio de Educación, Cultura y Deporte (Spain).

References

1. Beard, R.W., McLain, T.W., Nelson, D.B., Kingston, D., Johanson, D.: Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. *Proc. IEEE* **94**(7), 1306–1324 (2006)
2. Ollero, A.: Aerial robotics cooperative assembly system (arcas): first results. In: *Aerial Physically Acting Robots (AIRPHARO) Workshop, IROS 2012, Vilamoura, Portugal, 7–12 Oct 2012*
3. Merino, L., Caballero, F., Martínez de Dios, J.R., Maza, I., Ollero, A.: An unmanned aircraft system for automatic forest fire monitoring and measurement. *J. Intell. Robot. Syst.* **65**(1–4), 533–548 (2012)
4. Cobano, J.A., Martínez-de Dios, J.R., Conde, R., Sánchez-Matamoros, J.M., Ollero, A.: Data retrieving from heterogeneous wireless sensor network nodes using uavs. *J. Intell. Robot. Syst.* **60**(1), 133–151 (2010)
5. Gilmore, J.F.: Autonomous vehicle planning analysis methodology. In: *AIAAA Guidance Navigation Control Conference*, pp. 2000–4370 (1991)
6. Szczerba, R.J.: Threat netting for real-time, intelligent route planners. In: *IEEE Symposium Information, Decision Control*, pp. 377–382 (1999)
7. Hruschka, E.R., Campello, R.J.G.B., Freitas, A.A., De Carvalho, A.C.P.L.F.: A survey of evolutionary algorithms for clustering. *Trans. Sys. Man Cyber Part C* **39**(2), 133–155 (2009). <http://dx.doi.org/10.1109/TSMCC.2008.2007252>
8. Li, Y., Ang, K., Chong, G., Feng, W., Tan, K., Kashiwagi, H.: Cautocdevelopmental search and optimisation enabled computer automated control system design. *Int. J. Autom. Comput.* **1**(1), 76–88 (2007)
9. Chang Wook Ahn, R.S.R.: A genetic algorithm for shortest path routing, problem and the sizing of populations. *IEEE Trans. Evol. Comput.* **6**(6), 566–579 (2012)
10. Cobano, J.A., Conde, R., Alejo, D., Ollero, A.: Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties. In: *Proceedings of IEEE International Robotics and Automation (ICRA) Conference*, pp. 4429–4434 (2011)
11. Lavalle, S.M.: Rapidly-exploring random trees: a new tool for path planning. In: *Computer Science Department, Iowa State University. Technical Report TR 98–11* (1998)
12. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **30**, 1–76 (2011)
13. Pignaton, C.P.T.L.E., Morado, A.: Middleware support in unmanned aerial vehicles and wireless sensor networks for surveillance applications. *Stud. Comput. Intell.* **237**, 289–296 (2009)
14. Mitchell, H.L.P.D., Qiu, J., Grace, D.: Use of aerial platforms for energy efficient medium access control in wireless sensor networks. *Comput. Commun.* **33**(4), 500–512 (2010)
15. Teh, S.K., Mejias, L., Corke, P., Hu, W.: Experiments in integrating autonomous uninhabited aerial vehicles (uavs) and wireless sensor networks. In: *2008 Australasian Conference on Robotics and Automation (ACRA 08)*. The Australian Robotics and Automation Association Inc., Canberra (2008). <http://eprints.qut.edu.au/15536/>
16. Valente, J., Sanz, D., Barrientos, A., Cerro, J., Ribeiro, A., Rossi, C.: An air-ground wireless sensor network for crop monitoring. *Sensors* **11**(6), 6088–6108 (2011). <http://www.mdpi.com/1424-8220/11/6/6088>
17. Martínez-de Dios, J., Lferd, K., de San Bernab, A., Nez, G., Torres-Gonzalez, A., Ollero, A.: Cooperation between uas and wireless sensor networks for efficient data collection in large environments. *J. Intell. Robot. Syst.* **70**(1–4), 491–508 (2013). <http://dx.doi.org/10.1007/s10846-012-9733-2>
18. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. *J. ACM* **41**(4), 764–790 (1994)
19. Kuchar, J.K., Yang, L.C.: A review of conflict detection and resolution modeling methods. *IEEE Trans. Intell. Transp. Syst.* **1**, 179–189 (2000)
20. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. *J. Intell. Robot. Syst.* **57**(1–4), 65–100 (2010)

21. Prasanna, H.M., Ghosey, D., Bhat, M.S., Bhattacharyya, C., Umakant, J.: Interpolation-aware trajectory optimization for a hypersonic vehicle using nonlinear programming. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, San Francisco, USA, Aug 2005
22. Vela, A., Solak, S., Singhose, W., Clarke, J.-P.: A mixed integer program for flight-level assignment and speed control for conflict resolution. In: Proceedings of the 48th IEEE Conference on Decision and Control, pp. 5219–5226, Dec 2009
23. Pallottino, L., Feron, E., Bicchi, A.: Conflict resolution problems for air traffic management systems solved with mixed integer programming. *Int. Transp. Syst. IEEE Trans.* **3**(1), 3–11 (2002)
24. Bauso, D., Giarre, L., Pesenti, R.: Multiple uav cooperative path planning via neuro-dynamic programming. In: 43rd IEEE Conference on Decision and Control, pp. 1087–1092. Nassau, Bahamas, Dec 2004
25. Geiger, B.: Unmanned aerial vehicle trajectory planning with direct methods. Ph.D. dissertation, The Pennsylvania State University, Pennsylvania, USA (2009)
26. Vera, S., Cobano, J.A., Heredia, G., Ollero, A.: An hp-adaptative pseudospectral method for collision avoidance with multiple uavs in real-time applications. In: IEEE International Conference Robotics and Automation (ICRA), pp. 4717–4722. Hong-Kong, China, 31 May–7 June 2014
27. Spall, J.C.: Introduction to Stochastic Search and Optimization, 1st edn. Wiley, New York (2003)
28. Chakrabarty, A., Langelan, J.W.: Flight path planning for uav atmospheric energy harvesting using heuristic search. In: AIAA Guidance, Navigation and Controls Conference, Toronto, Canada, Aug 2010
29. Lamont, G.B., Slear, J., Melendez, K.: Uav swarm mission planning and routing using multi-objective evolutionary algorithms. In: IEEE Symposium on Computational Intelligence in Multicriteria Decision Making, pp. 10–20. Honolulu, Hawaii, USA, 1–5 April 2007
30. Conde, R., Alejo, D., Cobano, J.A., Viguria, A., Ollero, A.: Conflict detection and resolution method for cooperating unmanned aerial vehicles. *J. Intell. Robot. Syst.* **65**, 495–505 (2012). doi:[10.1007/s10846-011-9564-6](https://doi.org/10.1007/s10846-011-9564-6)
31. Alejo, D., Cobano, J.A., Heredia, G., Ollero, A.: Collision-free 4d trajectory planning in unmanned aerial vehicles for assembly and structure construction. *J. Intell. Robot. Syst.* **73**, 783–795 (2014)
32. Durand, N., Alliot, J.: Ant colony optimization for air traffic conflict resolution. In: Proceedings of the Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009), Napa, CA, USA (2009)
33. Xue, E.M., y Atkins, M.: Terminal area trajectory optimization using simulated annealing. In: 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, Jan 2006
34. Lavalley, S.M., Kuffner, J.J., Jr: Rapidly-exploring random trees: progress and prospects. In: Algorithmic and Computational Robotics: New Directions, pp. 293–308 (2000)
35. Alejo, D., Conde, R., Cobano, J., Ollero, A.: Multi-UAV collision avoidance with separation assurance under uncertainties. In: IEEE International Conference on Mechatronics, ICM 2009, pp. 1–6, April (2009)
36. LaValley, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006). <http://planning.cs.uiuc.edu/>
37. Akgun, B., Stilman, M.: Sampling heuristics for optimal motion planning in high dimensions. In: International Conferences on Intelligent Robots and Systems (IROS2011), pp. 2640–2645. San Francisco, USA, 25–30 Sept 2011
38. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *Syst. Sci. Cybern., IEEE Trans.* **4**(2), 100–107 (1968)
39. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. *IEEE Robot. Autom. Mag.* **19**(4), 72–82, Dec 2012. <http://ompl.kavrakilab.org>
40. Otte, M., Correll, N.: 1 c-forest: parallel shortest-path planning with super linear speedup. *IEEE Robot* **29**(3) 798–806 June 2013
41. Yang, K., Sukkarieh, S.: Planning continuous curvature paths for uavs amongst obstacles. In: Australasian Conference on Robotics Automation, Canberra, Australia (2008)

Multi-robot Surveillance Through a Distributed Sensor Network

Andrea Pennisi, Fabio Previtali, Cristiano Gennari,
Domenico D. Bloisi, Luca Iocchi, Francesco Ficarola,
Andrea Vitaletti and Daniele Nardi

Abstract Automatic surveillance of public areas, such as airports, train stations, and shopping malls, requires the capacity of detecting and recognizing possible abnormal situations in populated environments. In this book chapter, an architecture for intelligent surveillance in indoor public spaces, based on an integration of *interactive* and *non-interactive* heterogeneous sensors, is described. As a difference with respect to traditional, passive and pure vision-based systems, the proposed approach relies on a distributed sensor network combining RFID tags, multiple mobile robots, and fixed RGBD cameras. The presence and the position of people in the scene is detected by suitably combining data coming from the sensor nodes, including those mounted on board of the mobile robots that are in charge of patrolling the environment. The robots can adapt their behavior according to the current situation, on the basis of a

A. Pennisi (✉) · F. Previtali · C. Gennari · D. D. Bloisi · L. Iocchi · F. Ficarola ·
A. Vitaletti · D. Nardi

Department of Computer, Control, and Management Engineering,
Sapienza University of Rome, via Ariosto 25, 00185 Rome, Italy
e-mail: Pennisi@dis.uniroma1.it

F. Previtali
e-mail: Previtali@dis.uniroma1.it

C. Gennari
e-mail: Gennari@dis.uniroma1.it

D. D. Bloisi
e-mail: Bloisi@dis.uniroma1.it

L. Iocchi
e-mail: Iocchi@dis.uniroma1.it

F. Ficarola
e-mail: Ficarola@dis.uniroma1.it

A. Vitaletti
e-mail: Vitaletti@dis.uniroma1.it

D. Nardi
e-mail: Nardi@dis.uniroma1.it

© Springer International Publishing Switzerland 2015

A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604, DOI 10.1007/978-3-319-18299-5_4

Prey-Predator scheme, and can coordinate their actions to fulfill the required tasks. Experimental results have been carried out both on real and on simulated data to show the effectiveness of the proposed approach.

Keywords Mobile robots · Wireless sensor networks · Multi-robot systems · Multi-robot surveillance

1 Introduction

A critical infrastructure (CI) is a system which is essential for the maintenance of vital societal functions. Public areas, such as airports, train stations, shopping malls, and offices, are examples of CIs that can be a target for terrorist attacks, criminal activities or malicious behaviors. Usually, CIs are monitored by passive cameras with the aim of detecting, tracking, and recognizing objects of interest to understand and prevent possible threats.

However, traditional vision-based systems can result ineffective when dealing with realistic scenarios, since their passive sensors can fail in identifying and tracking an object of interest in a large environment, due to partial and total occlusions, changes in illumination conditions, and difficulties in re-identifying objects in different non-overlapping views. Moreover, a network of fixed passive sensors can be subject to malicious physical attacks [11].

1.1 Contributions of the Book Chapter

In this chapter, the problem of monitoring a populated indoor environment is faced by combining data coming from multiple heterogeneous fixed and mobile sensors. The term “populated” is used through the book chapter to denote an environment with presence of people. In our description, we do not take into account crowded or densely-populated environments. We describe the development of an architecture designed for the surveillance of a large scenario, where authorized personnel wear Radio Frequency Identification (RFID) tags and the environment is monitored by fixed RGBD cameras with RFID receivers and it is patrolled by multiple mobile robots, equipped with laser range finders and RFID receivers (see Fig. 1). Laser scans, RFID tag data, and RGBD images gathered by the distributed sensors are merged to obtain information about the position and the identity of people in the scene. Moreover, the robots coordinate their actions through a dynamic task assignment to fully cover the operational environment.

The architecture is conceived to work in a fully distributed fashion and to automatically raise alarms (possibly communicated to a central operational station) when abnormal conditions are detected. To this end, the developed architecture integrates different technologies. Although all the above technologies have been already

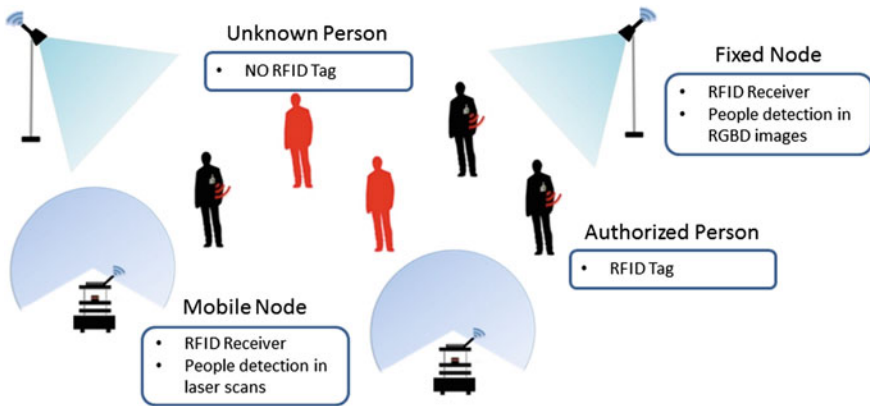


Fig. 1 The proposed architecture, combining mobile robots, fixed RGBD cameras, and RFID tags and receivers to monitor a populated environment

developed in previous works, their integration was not previously considered, in particular for surveillance applications. Moreover, an experimental analysis, carried out also in a real environment, shows the effectiveness of the implemented system.

The remainder of the chapter is organized as follows. Related work is analyzed in Sect. 2, while the definition of the problem is given in Sect. 3. The components of the architecture are described in Sect. 4 and the process of fusing the information coming from the different sensors is described in Sect. 5. The multi-robot coordination and the task assignment processes are detailed in Sect. 6. Results on both a real and a simulated environment are discussed in Sect. 7. Conclusions and future directions are drawn in Sect. 8.

2 Related Work

There exists a large literature about the problem of people detection in indoor environments by using fixed cameras. However, since a variety of factors, including illumination conditions, occlusions, and blind spots, limit the capacity of pure vision-based systems, it is possible to consider a combination of multiple heterogeneous sensors to achieve better results.

Approaches integrating multiple sensors can be divided into two main categories: (1) *interactive methods*, where each person has an active role during the detection process (e.g., by dressing an RFID tag) and (2) *non-interactive methods*, where the role of the person is passive and the analysis is carried out by the detection system only (e.g., a camera). In the rest of this section, some examples of interactive and non-interactive methods are described.

2.1 Interactive Methods

One of the first experiments about collecting information from a group of people in a physical real context is described by Hui et al. [8]: 54 individuals attending to a conference, dressed with an Intel iMote device consisting of a micro-controller unit (MCU), a Bluetooth radio and a flash memory, are considered. However, the choice of using Bluetooth does not allow for a fine-grained recording of social interactions, mainly because of the missing possibility of analyzing face-to-face interactions.

Multiple projects focusing on collecting data from social interactions are developed by the *SocioPatterns* collaboration. Partners participating in this collaboration have been the first to record fine-grained contacts by using active RFID sensors. This kind of devices allows to record face-to-face interactions within a range of 1.5 m. For example, Becchetti et al. [2] describe an experiment in which data coming from wireless active RFID tags worn by 120 volunteers moving and interacting in an indoor area are collected. The tags periodically broadcasts information about contacts with similar tags (i.e., whenever the person wearing the tag came close to another member of the volunteer group). Assuming that the subjects wear the tags on their chest and using very low radio power levels, contacts between tags are detected only when participants actually face one another, since the body effectively acts as a shield for the sensing signals. Thus, it is reasonable to assume that the experiment can detect an ongoing social contact (e.g., a conversation). *SocioPatterns* has made several installations in different social contexts, including conferences [1], hospitals [9], primary schools [16], and a science gallery [4], making some data sets publicly available on its website.¹

Experiments similar to the *SocioPatterns*' ones have been conducted deployed by Chin et al. [5], consisting in monitoring people wearing active RFID badges during a conference. The goal is to build a system that can find and connect people to each other. A remarkable result of the experiment is that, for social selection, more proximity interactions lead to an increased probability for a person to add another as a social connection.

While the above approaches target the analysis of social human behaviors, in this book chapter we investigate the use of data acquired from interactive tags for surveillance applications. Indeed, we aim at integrating the *SocioPatterns* sensing platform together with other sensing technologies, including laser range finders and RGBD cameras, to overcome the problems related to traditional automatic surveillance. It is worth noticing that a scenario in which (1) authorized personnel wear RFID tags and (2) other authorized actors (e.g., visitors, travelers, spectators) may have an RFID tag as well (e.g., included in a ticket or a passport or a boarding pass) is a quite plausible one. Airports, embassies, and theaters are examples of scenarios where interactive methods can be used.

¹<http://www.sociopatterns.org/datasets>.

2.2 Non-interactive Methods

Approaches in this category are based on passive sensors. Since the literature on vision-based systems is huge, we limit our description to existing approaches using technologies other than vision for addressing automatic surveillance. In the field of laser-based systems, Cui et al. [6] introduce a feature extraction method based on accumulated distribution of successive laser frames. A pattern of rhythmic swing legs is used to extract each leg of a person and a region coherency property is exploited to generate an efficient measurement likelihood model. A Kalman and a Rao-Blackwellized Monte Carlo data association (RBMC-DAF) filters are combined to track people. However, this approach is not effective for people moving quickly or partially occluded.

Xavier et al. [17] describe a feature detection system for real-time identification of lines, circles, and legs from laser data. Lines are detected by using a recursive line fitting method, while leg detection is carried out by taking into account geometrical constraints. This approach cannot handle scan data of a dynamic scene including moving people or not well separated structures.

A solution involving human-robot interaction is presented by Shao et al. [14]. Visual and laser range information are combined: Legs are extracted from laser scans and, at the same time, faces are detected from the images of a camera. A mobile robot uses the detection procedure (that returns the direction and the distance of surrounding people) to approach and to start interacting with humans. However, the swinging frequency is too low for people detection and tracking.

In the above cited papers, the main limitation concerns the problem of detecting multiple people. In most cases, the approaches can deal with well separated objects, but cannot be easily extended when multiple people are grouped together. In this book chapter, we propose an approach that can be used in a populated environment and that is suitable for monitoring groups of people. The method combines interactive and non-interactive heterogeneous sensors in order to overcome the problems of traditional vision-based systems.

Information coming from range finders, RFID receivers, and RGBD cameras are merged to obtain the position and the identity of people in the scene. Moreover, the actions of the robots are coordinated according to a dynamic task assignment algorithm, in order to have a dynamic monitoring range.

3 Problem Definition

The problem of monitoring a populated environment can be modeled as a *Prey-Predator* game. Indeed, considering the sensor nodes as predators and the objects to be monitored as preys, it is possible to formalize the surveillance task as follows: *A predator tries to catch preys and a prey runs away from predators.*

The game consists of preys and predators living in the same environment. It is usually defined as a game where both predators and preys have a score and any individual can gain or lose points over time. A metric distance is assigned to each prey and to each predator as the game score. The goal for each prey is to maximize its distance from the predators, while each predator aims at minimizing its distance from the preys. In our setting, the preys are the people moving in the monitored environment, while the predators are the sensor nodes that are used for detecting the presence and for estimating the position of a person. A sensor node is made of an RFID reader and other additional sensors, like an RGBD camera or a laser range finder. Moreover, some sensor nodes are mounted on mobile robots that navigate in the environment. For such a reason and for the presence of blind areas also, the portion of the environment that is currently observable can vary over time.

The monitoring task consists of identifying every person that does not wear an RFID tag by assigning her/him an *identity number* (ID). The goal of the monitoring task is achieved whenever a sensor node can detect the presence and the position of a person, determining if such a person is wearing or not an RFID tag. Formulating the surveillance task as a Prey-Predator game provides the following advantages: (1) In the case of a person leaving the monitored area and then re-entering later, the re-identification problem is not an issue, since if a person was labeled with an ID i before exiting the scene, when she/he re-enters the scene the system can use another ID j ($i \neq j$) and continue its process of determining if j is wearing or not an RFID tag; (2) The same performance metric defined for the Prey-Predator game can be used for evaluating our approach, providing quantitative results (see the experiments reported in Sect. 7).

4 Sensor Nodes

The proposed monitoring approach uses a combination of multiple heterogeneous fixed and mobile sensor nodes. Authorized people wear RFID tags of the type shown in Fig. 2a. Mobile nodes are robots equipped with a laser range finder and an RFID receiver (Fig. 2b), while fixed nodes are made of RGBD cameras to grab visual 3D information and RFID receivers that are mounted near the camera (Fig. 2c). The communication between mobile and fixed sensors is achieved by using TCP/IP over a wireless network. This is a feasible solution, since the size of exchanged messages among nodes is quite small (up to 1 kb) and the possibility to either lost a message or receive a delayed message is negligible.

4.1 RFID Tags and Receivers

The two main entities of our sensing platform, designed and developed by the SocioPatterns research collaboration, are the OpenBeacon active RFID tags (Fig. 2a)

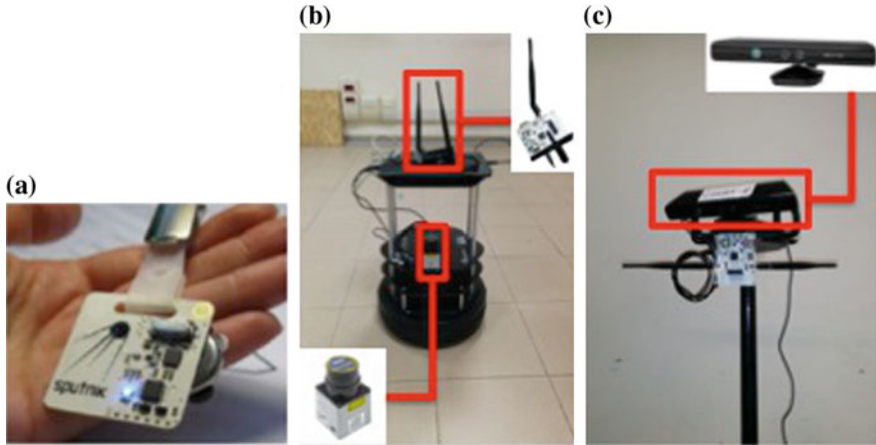


Fig. 2 **a** RFID tag. **b** Turtlebot robot equipped with a laser range finder and an RFID receiver. **c** Fixed RGBD camera with RFID receiver

and the OpenBeacon Ethernet reader (top right in Fig. 2b). The tags are electronic wireless badges equipped with a PIC16 micro-controller (MCU) and an ultra low power radio frequency transceiver. The MCU has a total SRAM of 256 bytes and can work up to 8 MHz of frequency, while the transceiver has very low energy consumptions: 11.3 mAh in transmission at 0 dBm of output power and 12.3 mAh in reception at 2 Mbps of air data rate. They are powered by batteries ensuring a lifetime of more than two weeks and are programmed to periodically broadcast beacons of 32 bytes at four different levels of signal strength: 0, -6 , -12 , -18 dBm. Every beacon contains the tag identifier, the information about the current signal strength, and other fields useful for debugging. Similarly, the RFID reader has a transceiver as well and an omni-directional covering range of 10 m.

The whole sensing platform is designed to allow the RFID receivers to collect the data sent by each tag via the wireless channel. In our scenario, a receiver is mounted on each robot and it is used to read the signal strength and the ID of a tag, in order to establish if a person detected in the environment is actually wearing a tag. All data collected by RFID readers are forwarded to a central logging server,² that stores all messages in log-files. Each record contains information about the tag whom sent the packet, including its ID, the signal strength, the sequence number and the IP of the reader that collected the corresponding message.

²OpenBeacon Logger. <https://github.com/francesco-ficarola/OpenBeaconLogger>.

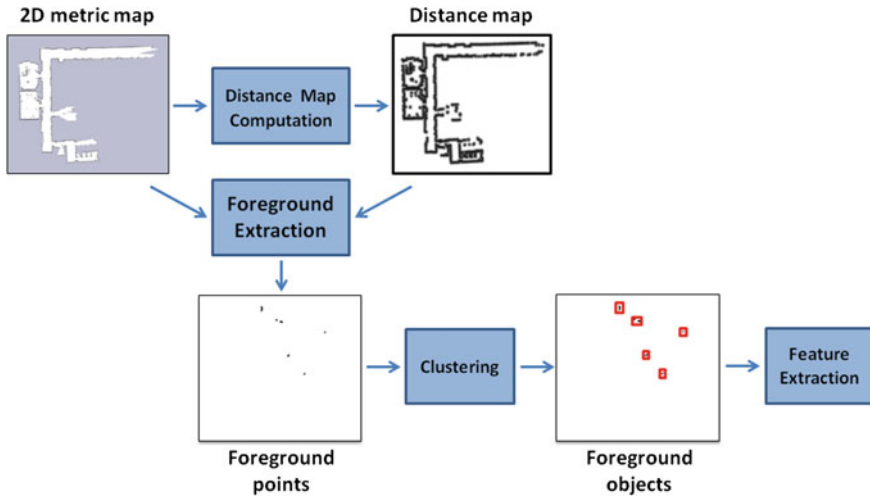


Fig. 3 People detection using the laser range finder

4.2 Laser Range Finders

The mobile sensor node is composed of a Turtlebot³ equipped with a range finder and an RFID receiver (Fig. 2b). Multiple robots are involved in the task of patrolling the environment. Each robot has a 2D metric map of the environment, that is built off-line using the ROS gmapping tool.⁴ Furthermore, each robot can be considered always well-localized on the 2D metric map by using the ROS implementation of the AMCL localization method.⁵ Person detection is carried out by means of a *distance map*, indicating the probability that a given point in the current laser scan belongs to the metric map. By comparing the distance map with the metric map it is possible to extract the foreground objects, i.e., sets of points in the distance map that are far enough from the metric map points. From each foreground object the following features are extracted: the number of its points, their standard deviation, a bounding box, and the radius of the minimum enclosing circle (see Fig. 3). Then, the features are sent as input to an Ada-Boost based person classifier, trained with about 1800 scans.

People tracking relies on the particle filter algorithm called *PTracker*, that is described in Sect. 5. Data association is used to determine the relationship between observations and tracks, and multiple hypotheses are maintained when observations may be associated to more than one track. Finally, each track is combined with the signal detected by the RFID receiver mounted on each robot, in order to verify if a person is wearing the RFID tag.

³<http://www.turtlebot.com/>.

⁴<http://wiki.ros.org/gmapping>.

⁵<http://wiki.ros.org/amcl>.

4.3 RGBD Cameras

The Microsoft Kinect (version 1.0) has been used as RGBD camera. Kinect sensor supplies an RGB image with a resolution of 640×480 and a frame rate of 30 frames per second. 3D information are received in the form of a 11-bit depth image. Both color and depth information are used for computing an accurate foreground detection. RGB and depth data are stored for each captured frame.

A statistical approach, called Independent Multimodal Background Subtraction (IMBS) [3], is used to create the background model, that is updated every 15 seconds for dealing with illumination changes. The obtained foreground mask is used as starting point for a 3D clustering step.

Let \mathfrak{B} denote the set of 3D points that corresponds to the 2D points belonging to the blobs in the foreground mask and \mathcal{C} denote all the 3D points of the point cloud generated from the depth data provided by the Kinect ($\mathfrak{B} \subset \mathcal{C}$). In order to improve the detection results, all the 3D points $\in \{\mathcal{C} \setminus \mathfrak{B}\}$ having a distance < 0.01 m from the points in \mathfrak{B} are recursively added to \mathfrak{B} itself. Then, to filter out possible false positives, all the blobs with a maximum height < 1.2 m are discarded, while the others are considered as valid observations. Finally, the 3D positions of the valid blobs are computed by estimating their Euclidean distance from the cameras. Indeed, since the positions of the cameras monitoring the environment are known, people can be localized on the 2D metric map by averaging the 3D points belonging to the their blobs and calculating the distance of the average point from the camera. The above described steps are summarized in Fig.4.

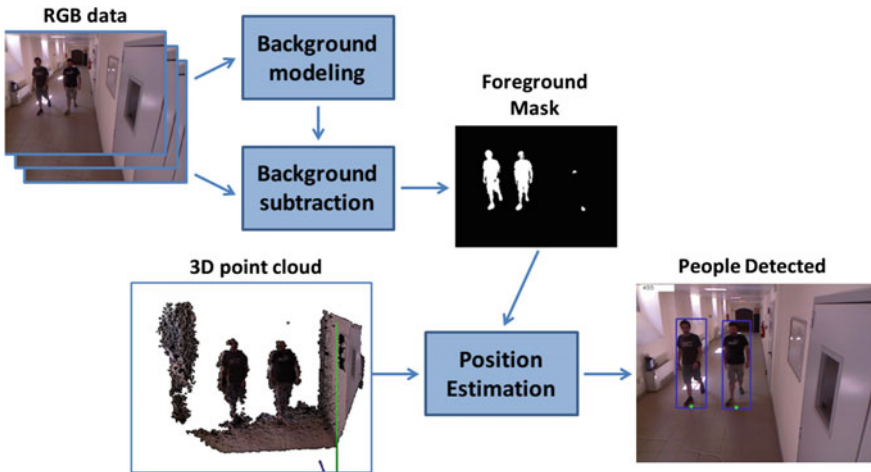


Fig. 4 People detection using a RGBD camera

5 Data Fusion

Information coming from fixed and mobile sensor nodes needs to be merged. The data fusion process is made of two phases: (1) Obtaining tracks by fusing visual and laser data and (2) Merging the tracks with RFID receiver information.

In the first phase, a multi-object particle filter approach, called PTracker, has been used in order to fuse and track the observations extracted from RGBD and laser data. A particle filter-based tracker maintains a probability distribution over the state of the object being tracked, keeping information about position, scale, color, direction and velocity of the object. Particle filters represent this distribution as a set of weighted samples (particles). Each particle represents a possible instantiation of the state of the object and it is a guess representing one possible position of the object being tracked. The set of particles contains more weight at locations where the object being tracked is more likely to be. This weighted distribution is propagated through time the Bayesian filtering equations, and the trajectory of the tracked object is determined by taking the particle with the highest weight or the weighted mean of the particle set at each time step. A detailed description of the data fusion method is available at <http://www.dis.uniroma1.it/~previtali/downloads/DataFusion.pdf>. The output of this phase is a set $S_t = \{o_t^1, \dots, o_t^n\}$ containing all the observations o_t^i at time t , $1 \leq i \leq n$, where n is the total number of the observations.

In the second phase, S_t is merged with the information coming from the RFID receivers at time t , the set $U_t = \{id_t^1, \dots, id_t^k\}$, where id_t^k is a triple $\langle t, p, r \rangle$, with t being the identification number of the tag, p the pose of the receiver that detects the tag t , and r the detection range of the receiver. An observation $o_t^i \in S_t$ is associated to a triple $(id_j^t = \langle t_j, p_j, r_j \rangle) \in U_t$ if all the particles of o_t^i (computed by PTracker) are included in the circular range having radius r_j and center p_j . After the merging phase, three different outputs can be generated:

1. $z_h^t = \langle o_t^i, id_j^t \rangle$ where o_t^i has been merged with id_j^t ;
2. $z_h^t = \langle o_t^i, ? \rangle$ where no RFID data have been associated with the track o_t^i ;
3. $z_h^t = \langle ?, id_j^t \rangle$ where no track information can be assigned to the detected RFID data id_j^t .

6 Multi-robot Surveillance

In our architecture, multiple robots are responsible for patrolling the environment, meaning that a team of different robotic agents have to be coordinated and controlled. This can be done by adopting different coordination strategies and control approaches, in order to plan the robots' behavior. We propose a novel approach based on a distributed coordination and a hybrid control, that makes use of a variant of the multi-robot Petri Net Plans [18].

6.1 *Distributed Coordination*

Coordination strategies for a team of robots can be divided into two categories: (1) Centralized methods, where an agent sends commands to other robots, and (2) Distributed approaches, where each agent decides its task and it shares information with the team.

In centralized solutions, the whole planning task is assigned to a single agent (central unit) that is responsible to calculate the next task for each robot. This must be done in a very short time, due to real-time constraints. Furthermore, the central unit represents a weak point, since in case it crashes, this will affect the functionality of the entire system.

Adopting a distributed approach, it is possible to deal with the above issues. The idea is to make each agent acting independently from each other, by using only local knowledge about the environment. An agent can collaborate with its neighbors in order to divide the task into sub-problems or to work together to achieve the defined goals.

Even if a distributed coordination increases the computation costs, due to the need of managing the necessary coordination messages, it allows to split the costs among all the team. Indeed, each robot performs a smaller amount of computation with respect to the whole computation load required to a single central node. The communication between agents in a distributed processing is greatly reduced as well, since it is not necessary to exchange information about perception, thus reducing the transmissions to a simple exchange of lightweight coordination messages. In addition, the reaction to external events is faster, since events are managed locally, without the need of waiting for instructions from a central global coordinator. Finally, a distributed execution is more robust to failures. Indeed, if an agent becomes unreachable, it can be replaced by another one without affecting the capability of the system.

6.2 *Petri Nets Plans*

We propose a distributed architecture with hybrid control and centralized planning. Using a centralized plan means that a plan is generated from a supervisor (an agent or a user) and it is sent in a distributed fashion to all the robots. Petri Nets Plans (PNP) [18] is a framework based on Petri Nets, conceived for designing, writing, executing, and debugging plans. The use of PNP allows a clear distinction between action specification and their implementation as well as a formal specification of plans, which permit to implement reasoning and to verify procedures.

Ziparo et al. [18] have extended the PNP framework to multi-robot systems. The extension uses a shared plan, that provides each agent with a plan created in a centralized manner, and with the model to run it in a distributed fashion. This approach allows to execute a set of PNPs, created by a shared multi-robot plan, for a single-robot, without the need of a central coordinator. The correctness of the distributed

execution with respect to the multi-robot PNP is enforced by using the communication primitives *send*(ID), *receive*(ID) and *sync*(ID, ID'), where ID and ID' are unique identifiers for the state of execution of single-robot plans. The primitives are modeled as single-robot ordinary non-instantaneous actions and represent communication acts and they are used to define three operators for coordinating the plans for each single robot:

- **Hard Synchronization.** It synchronizes in time two single-robot plans and it allows for information sharing among them, through the communication of ID_s and ID_r which encode the state of execution for the plan of agent s and agent r , respectively.
- **Soft Synchronization.** It defines a precedence relation among two actions of two different robots.
- **Multi-robot Interrupt.** It allows for relating interrupts between the actions of two robots s and r . Since each robot has a receiving thread, when a sensing action launches an interrupt on s , it send a message to r , which starts an interrupt as well.

6.3 Hybrid Control

A robot decides the type of action to perform as the result of the perceived information from the environment. It contains two procedures, a reactive and a deliberative one, with an interface that is responsible for connecting them. The reactive procedure is used for handling situations that require an immediate reaction of the robot, such as avoiding obstacles or sending an alarm in case of intruders. The deliberative procedure is used for long-term decisions, like planning a trajectory. This two-level architecture has the advantage of increasing the reactivity of the system, having a dynamic and very responsive control. It is worth noting that, designing the interface between the two layers is not trivial.

The PNP formalism is used to manage both the deliberative and reactive control. In [18] the global plan is converted into individual single-robot plans. In such a way, a manual or an automatic rewriting of the single plans is needed. However, the individual plans must be distributed to each robot, making the operations of changing and debugging a plan rather complex.

We propose a different approach. Each agent is equipped with an execution model that can interpret and execute the original PNP. Then, the plan interpreter manages the various actions by implementing those addressed to it and by handling the actions which pertain to different agents, through communication primitives. The main difference with [18] lies in the PNP executor. Indeed, in [18] each received command is interpreted by using operators and primitives both defined in PNP, then the appropriate commands are sent to the robot in order to complete the required tasks (see Fig. 5a).

Our approach includes, instead, two new software modules: (1) the Coordinator and (2) the Robot ID Recognizer. The Coordinator is not a specific robot, but a

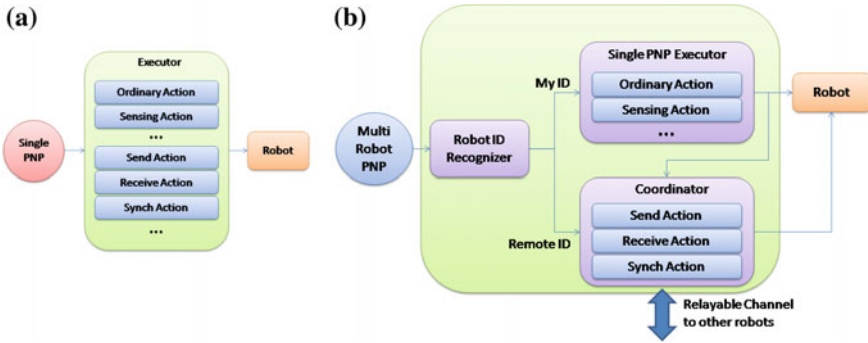


Fig. 5 **a** Architecture for the PNP Executor proposed in [18]. **b** Our modified architecture for the PNP Executor

software procedure running on all the robots in the team. Both modules are designed to work in a distributed asynchronous fashion. We assume that all the information are sent and received asynchronously between the robots and that some robots can be unable to receive all the available information due to lost packets.

The PNP executor is the same as the one in the single-robot case, which uses only the PNP primitives and operators. In the PNP plan, by following the rules of labeling, an action is preceded by the ID of the robot that has to perform it. First of all, the action is passed to the Robot ID Recognizer, which controls whether the action has to be executed by the local agent or by one of the remote robots. If it is the case for the local agent, the action is passed to the single PNP executor. Instead, if the action is for a remote robot, it is passed to the local coordinator module which transforms the action into a communication primitive, in order to synchronize the evolution of the local plan with the execution state of the remote robots. This can be done since the coordinator informs the other robots about the ending of the local action (see Fig. 5b), thus allowing to synchronize the plans of all the agents. A copy of the multi-robot plan is stored by all robots, and all the PNP executors are synchronized. In such a way, if no communication errors occur, each agent exactly knows the execution state of all the remote agents.

6.4 Implementation

For the low-level control, we use the Robotic Operative System (ROS),⁶ which is a flexible framework for robotic infrastructures equipped with a collection of tools, libraries, and conventions aiming at simplifying the creation and management of robotic platforms. The tools are arranged in nodes, that can be integrated with other nodes to compose a complex architecture.

⁶<http://www.ros.org>.

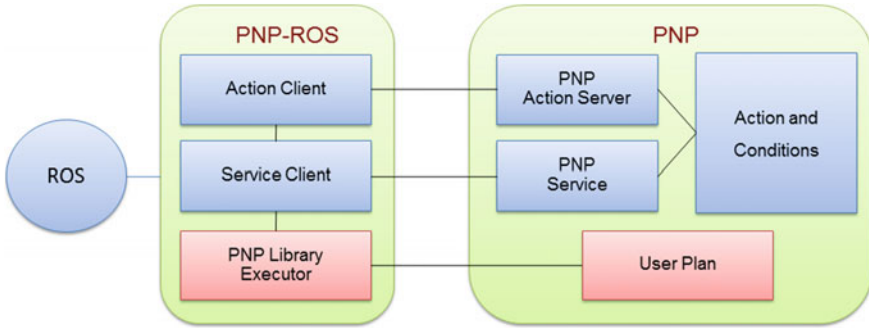


Fig. 6 Scheme of the PNP-ROS bridge

In order to interact with ROS, a node has been built for interfacing the external libraries. The scheme of the bridge from PNP to ROS is reported in Fig. 6. The Action Client asks the Action Server to execute a given action, while the Service Client asks the PNP Service to evaluate the firing conditions. The PNP Service is user-defined and it maintains the state of the system giving a response (i.e., a value *true* or *false*) about a condition. When PNP-ROS sends a request for an action, the Robot ID Recognizer (see Fig. 5b) pre-processes the request by checking if the action has to be sent to the local agent (and therefore it will be performed), or if the action has to be sent to a remote robot. In the last case:

- If the request is for a local action, the Action Server launches a new communication thread involving the other ROS modules in order to accomplish the task. When the action is finished, the boolean state variable is set to *true* and the PNP Service can respond with a positive value. This means that the PNP library Executor knows that the action is terminated and it can send an “ActionFinished” message to all the remote agents and proceed with another plan.
- If the address of the action is a remote agent *id*, the Action Server launches a new communication thread with a primitive *receive(id)*. This is a blocking function and thus the thread stops its execution waiting for a message. When the remote agent finishes its action, it sends an “ActionFinished” message and the thread can resume.

Using the above described protocol, each agent can directly use the multi-robot original plan, while maintaining the information about the state of execution of the plan for the other agents.

6.5 Example of PNP Execution

Figure 7 shows an example of a simple PNP execution. Two sensor nodes are monitoring the environment: Node0 is a fixed camera with an RFID receiver, while Robot1 is a mobile robot equipped with a laser range finder and an RFID receiver. Both nodes

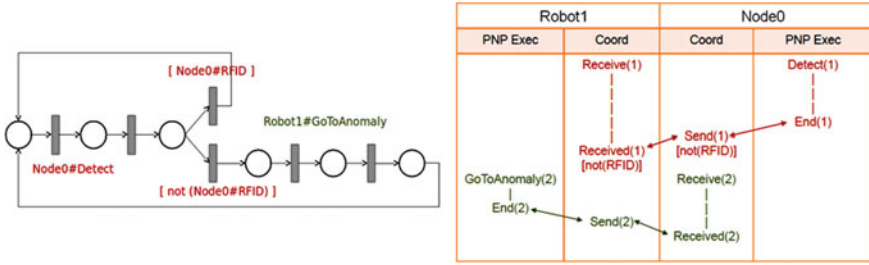


Fig. 7 Example of PNP execution

receive a request for the action “Node0#Detect” meaning that Node0 is required to check if a person is in its field of view. Robot1 understands that the action is for a remote agent, thus it sends the request to the Coordinator, that transforms the action in a primitive *receive*. When Node0 detects the presence of a person, it completes its action and sends a message, containing the information about the presence of an authorized person or not, to Robot1. In the example shown in Fig. 7, the person does not wear an RFID tag, so Robot1 is notified that it has to check the possible abnormal situation (action “Robot1#GoToAnomaly”).

6.6 Dynamic Task Assignment

The robots in the team must work together on the current task, coordinating their actions and efficiently sharing the workload to maximize the overall task performance. This is a complex goal, since the robots operate in a dynamic environment and the perceptions can be noisy.

In order to deal with the coordination problem in a real scenario, we adopted a solution based on a greedy algorithm [12, 15] and the Prey-Predator game formalization (see Sect. 3). A Dynamic Task Assignment (DTA) process is responsible for assigning a prey to a predator. Such an assignment is unique, meaning that a predator cannot chase two or more preys. A predator creates a new *bid* each time it sees a prey (Fig. 8a). A bid describes its estimates of the expected information gain and costs of traveling to various locations for catching the prey. Bids, that are the same for both mobile and fixed sensors, are asynchronously sent to all the predators (Fig. 8b) and the DTA algorithm makes the assignment on the basis of the current bids (Fig. 8c). The tracking performance of a predator increases at the decrease of its distance from a prey, thanks to the higher quality of the received sensor data. During the chasing, a predator could change the prey to chase: To handle this situation, the DTA algorithm assigns the prey no longer chased to another predator.

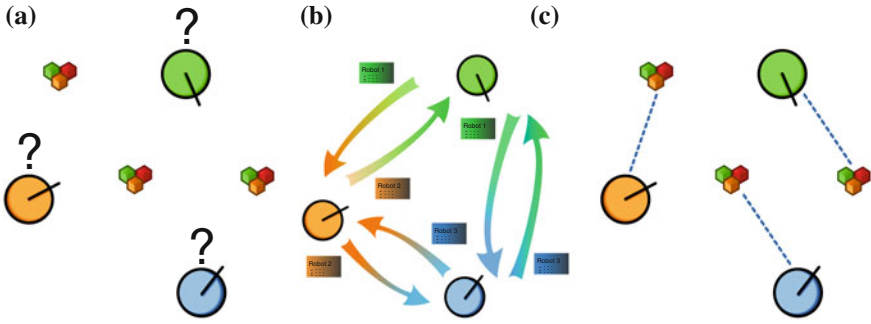


Fig. 8 Dynamic task assignment (DTA). **a** Predators do not yet know which prey chase. **b** Predators exchange their bids. **c** The DTA algorithm assigns at each predator the best prey to chase

7 Experimental Evaluation

Experimental results has been computed both in a real scenario and by using a simulator. The experiments carried out in the real scenario have been used to generate the error models for the sensors in the network nodes (a model for the RGBD camera and one for the laser range finder). The models are very useful for obtaining realistic results in the simulated environment, that are then used to quantitatively evaluate the effectiveness of the proposed architecture.

7.1 Experiments with Real Data

Experiments with real data have been performed with two purposes: (1) To generate the error models for the sensor nodes, and (2) To demonstrate the overall feasibility of the developed system.

The first set of experiments are thus focused on determining the error models of the sensors used for people detection: an RGBD camera (a Kinect sensor) and a laser range finder (a Hokuyo UTM-30LX). The setup is given by two fixed sensor nodes each including one of the two sensors and a person standing at a variable distance d from the sensor nodes (see Fig. 9). Four runs for each considered distance (ranging from 1 to 4 m) have been considered. The obtained results are reported in Table 1. As expected, the accuracy of the laser-based method is higher than the one of the RGBD-based technique, and the errors increase with the distance, for both the laser and the RGBD camera. The results allow to determine a suitable error model for the sensors involved in the architecture.

Moreover, when considering a mobile sensor (i.e., an RGBD camera or a laser mounted on a robot), the error in the self-localization routine carried out by the robot must be taken into account, since it can influence the detection accuracy. To this end, we performed a set of preliminary tests on different Turtlebot robots in

Fig. 9 Laser and visual data are merged using the floor as a common reference frame

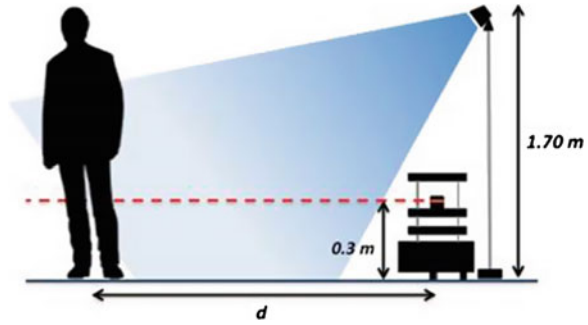


Table 1 Results in the real scenario

Sensor type	Real distance (m)	Detected distance (m)	Error (m)
Kinect	1	1.441	0.441
Laser	1	1.029	0.029
Kinect	2	2.404	0.404
Laser	2	2.040	0.040
Kinect	3	3.464	0.464
Laser	3	3.068	0.068
Kinect	4	4.533	0.533
Laser	4	4.066	0.066

order to calculate their localization error. We used the well-known approach by Fox et al. described in [7], obtaining a localization error in the range between 8 cm and 16 cm. The computed error models (sensors + localization) are used as input for the simulated experiments (described below) to obtain realistic observations during the simulations.

The second set of experiments with real data has been performed to show the effectiveness of the entire approach. Here we do not collect quantitative measures, but just demonstrate the whole architecture running. Some videos showing the experiments are available in [13] and some snapshots are reported in Fig. 10. The behavior of the system is the following. Whenever the fixed sensor node detects a person, a Turtlebot equipped with a laser range finder and an RFID reader is sent in that location, in order to verify the status of the person (i.e., if she/he is wearing or not the RFID tag) and to report the anomaly if it is the case (see Fig. 11). Otherwise, if the system does not detect anomalies, sends a message to the robot which continues patrolling the environment.

Finally, we measured the computational speed of the entire system processing in terms of frames per second (FPS) on live data coming from the sensors, using an Intel Core i5-3210M 2.50 GHz (2 cores), 4 GB RAM and a virtual machine with a



Fig. 10 Real experiment: **a** fixed node composed by a Kinect and a RFID receiver, **b** two RFID tags, **c** a Turtlebot robot equipped with a laser range finder

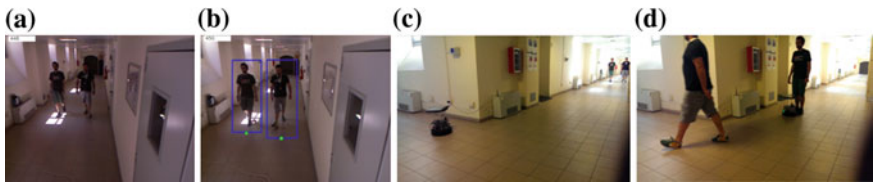


Fig. 11 Experiment with real data. **a** A person is wearing the RFID tag while the other one are not, **b** the fixed sensor detects the two people, identifies only one RFID tag and, **c** sends the coordinates to the mobile robot, **d** the robot stops in front of the person that is not wearing the RFID tag

Table 2 Computational speed for the RGBD detection module running with a single camera

Frame size	FPS (2 cores)	FPS (virtual machine, 2 cores)
320 × 240	25	20
640 × 480	23	18

simulated processor 2.00 GHz (2 cores), 4 GB RAM. The results are shown in Table 2, demonstrating that the proposed approach is suitable for real-time applications with commercial CPUs and even in the case of using a virtual machine.

7.2 Experiments in a Simulated Environment

The goal of the experimental evaluation on simulated data is to quantitatively evaluate the performance of our method. We run all the experiments by using the simulator Stage. In Stage, both the sensor nodes and the people are represented as robotic agents. The estimation of the position of the simulated people (i.e., the implementation of

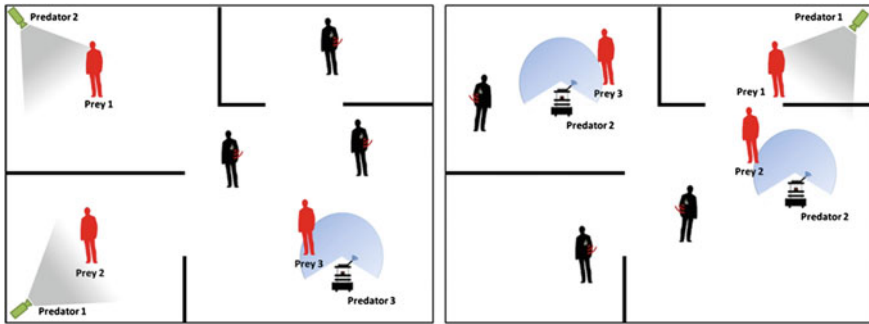


Fig. 12 The simulated environment in stage

the virtual sensors) is obtained by generating observations with the addition of an error calculated accordingly to the error model of the real sensors calculated in the experiments discussed above. Moreover, to have a realistic simulation, we adopted a realistic model of the people, the same model of the robots as well as the same field-of-view of the sensors as in the real scenario.

Figure 12 shows two screen-shots from an experiment in which three sensor nodes (i.e., predators) are chasing moving people without an RFID tag (i.e., preys). People with tags are no more chased once detected. The experiment has been carried out by launching multiple runs, changing every time the initial positions and the type of the sensors (fixed or mobile) and the starting positions of the people with and without tags. The average error has been calculated by using Eq. 1, while the standard deviation by using Eq. 2:

$$avg = \sum_{t=1}^n \sum_{i=1}^k \frac{1}{k} \frac{\sum_{j=1}^m \|e_{i,j}^{(t)} - g_j^{(t)}\|}{m} \quad (1)$$

$$std. dev. = \sum_{t=1}^n \sum_{i=1}^k \frac{1}{k} \frac{\sum_{j=1}^m \|e_{i,j}^{(t)} - avg_i^{(t)}\|}{m} \quad (2)$$

where $e_{i,j}^{(t)}$ is the j th estimation performed by the robot i at time t , $g_j^{(t)}$ is the ground-truth position provided by the simulator of the object j at time t , m is the number of estimations performed by the robot i at time t , n is the duration in seconds of the experiment and k is the number of robots (i.e., predators) involved in the experiment. The results obtained during the simulations are reported in Table 3: The low value of the standard deviation demonstrates a remarkable reliability of the proposed approach.

We also quantitatively measured the performance of the tracking module. To this end, we used the well-known CLEAR MOT [10] metrics MOTA and MOTP. MOTA (Multiple Object Tracking Accuracy) measures the accuracy and MOTP (Multiple

Table 3 Results in the simulated environment

Run #	Prey-predator distance (avg. \pm std. dev.) (m)	Run #	Prey-predator distance (avg. \pm std. dev.) (m)
1	0.81 \pm 0.13	6	0.64 \pm 0.17
2	1.22 \pm 0.21	7	0.79 \pm 0.31
3	0.83 \pm 0.15	8	1.18 \pm 0.35
4	1.43 \pm 0.08	9	1.03 \pm 0.22
5	1.38 \pm 0.13	10	1.39 \pm 0.28

Object Tracking Precision) calculates the precision of the tracking algorithm. MOTA results give a measure of how good the tracking algorithm can keep connect the object identities over time, while MOTP results are useful for evaluating the difference between the bounding box provided by the tracking algorithm and the minimum bounding box containing the tracked person. MOTA is defined as:

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t) + c_s(ID-SWITCHES_t))}{\sum_{t=1}^{N_{frames}} N_G^{(t)}} \quad (3)$$

where, after computing the mapping for frame t , m_t is the number of misses, fp_t is the number of false positives, $ID-SWITCHES_t$ is the number of ID mismatches in frame t considering the mapping in frame $(t - 1)$, and $N_G^{(t)}$ is the number of objects present in frame t . The values for the weighting functions have been set to $c_m = c_f = 1$ and $c_s = \log_{10}$.

To obtain the precision score, we calculated the spatio-temporal overlap between the reference tracks and the output tracks of our method. MOTP was defined as:

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}^{(i)}} \left[\frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|} \right]}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}} \quad (4)$$

where N_{mapped} refers to the mapped system output objects over an entire reference track taking into account splits and merges, and N_{mapped}^t is the number of mapped objects in the t th frame.

Table 4 reports MOTA and MOTP values for all the experiments that have been carried out. The results show that the integration of data coming from heterogeneous sensor nodes composed of active RFID tags, RGBD cameras, and mobile laser range finders can be used to deal with the problem of monitoring a populated environment. A more accurate experimental analysis for measuring false positive/false negative rates in different situations and integration with other techniques (e.g., vision) would further improve the assessment of the quality of the system.

Table 4 Results of the tracking method in the simulated environment

Experiment	MOTA	MOTP	Experiment	MOTA	MOTP
1	0.95	0.85	6	0.91	0.87
2	0.97	0.90	7	0.95	0.89
3	0.96	0.88	8	0.95	0.91
4	0.92	0.91	9	0.97	0.93
5	0.99	0.95	10	0.98	0.92

8 Conclusions

Integrating multiple technologies for surveillance applications is an important and necessary step towards the developing and deploying of effective systems. In this book chapter we describe an architecture and several techniques used for integrating heterogeneous fixed and mobile sensor nodes in order to determine the presence and the position of people in an indoor environment. Different technologies (RFID tags, laser range finders, and RGBD cameras) are combined through a distributed data fusion method, which is robust to perception noise and is scalable to multiple heterogeneous sensors. The reported experimental results, obtained both with real and simulated data, show the feasibility of the approach and the overall capabilities of the architecture. Automatic monitoring and detection of abnormal activities are possible and performance in this task can be good enough for an actual deployment. However, additional work must be done in order to make the techniques more precise and more robust.

A potential extension for the approach described in this book chapter consists in adding more types of sensors to the network, such as microphones, GPS receivers, and active RFID tags providing signal strength data. Moreover, the simulated scenario can be enriched by generating realistic error models specific to those additional sensors. In order to improve the multi-sensor architecture designed in this book chapter, the system can be tested by end-users to evaluate the usability and the feasibility of the proposed approach.

Acknowledgments This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research and it has been carried out in cooperation with the SocioPatterns collaboration (www.sociopatterns.org).

References

1. Barrat, A., Cattuto, C., Szomsoz, M., den Broeck, W.V., Alani, H.: Social dynamics in conferences: analysis of data from the live social semantics application. In: Proceedings of the 9th International Semantic Web Conference, pp. 17–33 (2010)
2. Becchetti, L., et al.: Population protocols on real social networks. In: Proceedings of the 5th Workshop on Social Network Systems, pp. 15:1–15:2 (2012)

3. Bloisi, D.D., Iocchi, L.: Independent multimodal background subtraction. In: Proceedings of the 3rd International Conference on Computational Modeling of Objects Presented in Images: Fundamentals, Methods and Applications, pp. 39–44 (2012)
4. Van den Broeck, W., Quaggiotto, M., Isella, L., Barrat, A., Cattuto, C.: The making of sixty-nine days of close encounters at the science gallery, Leonardo, pp. 285–285 (2012)
5. Chin, A., Xu, B., Wang, H., Wang, X.: Linking people through physical proximity in a conference. In: Proceedings of the 3rd International Workshop on Modeling Social Media, pp. 13–20 (2012)
6. Cui, J., Zha, H., Zhao, H., Shibasaki, R.: Laser-based detection and tracking of multiple people in crowds. *Comput. Vis. Image Underst.* 300–312 (2007)
7. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: efficient position estimation for mobile robots. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 343–349 (1999)
8. Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Pocket switched networks and the consequences of human mobility in conference environments. In: Proceedings of ACM SIGCOMM First Workshop on Delay Tolerant Networking and Related Topics, pp. 244–251 (2005)
9. Isella, L., Romano, M., Barrat, A., Cattuto, C., Colizza, V., Van den Broeck, W., Gesualdo, F., Pandolfi, E., Ravá, L., Rizzo, C., Tozzi, A.E.: Close encounters in a pediatric ward: measuring face-to-face proximity and mixing patterns with wearable sensors. In: PLoS ONE, p. e17144 (2011)
10. Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., Zhang, J.: Framework for performance evaluation of face, text, and vehicle detection and tracking in video: data, metrics, and protocol. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(2), 319–336 (2009)
11. Liu, C., James, T.Y.: An analysis of dos attacks on wireless lan. In: *Wireless and Optical Communications* (2006)
12. Palmer, D., Kirschenbaum, M., Murton, J., Zajac, K., Kovacina, M., Vaidyanathan, R.: Decentralized cooperative auction for multiple agent task allocation using synchronized random number generators. In: Proceedings of International Conference on Intelligent Robots and Systems (IROS), vol. 2, pp. 1963–1968 (2003)
13. Pennisi, A.: Multi-robot surveillance through a distributed sensor network experiments. <http://www.dis.uniroma1.it/pennisi/scij/>
14. Shao, X., Zhao, H., Shibasaki, R., Shi, Y., Sakamoto, K.: 3d crowd surveillance and analysis using laser range scanners. In: International Conference on Intelligent Robots and Systems (IROS), pp. 2036–2043 (2011)
15. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: *AAAI/IAAI*, pp. 852–858 (2000)
16. Stehlé, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J.F., Quaggiotto, M., Van den Broeck, W., Régis, C., Lina, B., Vanhems, P.: High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE* **6**(8), e23176 (2011)
17. Xavier, J., Pacheco, M., Castro, D., Ruano, A.: Fast line, arc/circle and leg detection from laser scan data in a player driver. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3930–3935 (2005)
18. Ziparo, V.A., Iocchi, L., Nardi, D., Palamara, P.F., Costelha, H.: Petri net plans: a formal model for representation and execution of multi robot plans. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 79–86 (2008)

Part II
Data Fusion, Localization
and Mapping

Exploiting Multi-hop Inter-beacon Measurements in RO-SLAM

A. Torres-González, J. Ramiro Martínez-de Dios and A. Ollero

Abstract This chapter presents a Range Only (RO) Simultaneous Localization and Mapping (SLAM) scheme that integrates multi-hop inter-beacon range measurements. While few SLAM schemes use inter-beacon measurements, to the best of our knowledge, none of them integrates multi-hop inter-beacon measurements. In our scheme the robot gathers inter-beacon measurements with configurable hop number so that it can integrate measurements between beacons far beyond the robot's sensing range. This chapter analyzes the impact of integrating in SLAM inter-beacon measurements with different hop numbers and evaluates its performance and robustness to measurement and odometry noise levels. It also validates its results in real experiments. It shows that the advantages of using inter-beacon measurements increase with measurements with higher hop numbers.

Keywords Robot-sensor network cooperation · SLAM

1 Introduction

This work is motivated by schemes of robot-sensor network cooperation where sensor nodes (beacons) are used as landmarks for RO-SLAM. The idea of building a map without any a priori knowledge and at the same time keeping track of the robot location is very appealing in a high number of applications.

This work was supported by the EC-SAFEMOBIL (European Commission ICT-2011-288082), CLEAR (DPI2011-28937-C02-01) and the Ministerio de Educación y Deportes FPU Program.

A. Torres-González · J.R. Martínez-de Dios · A. Ollero (✉)
Robotics, Vision and Control Research Group, University of Seville, Camino de Los Descubrimientos S/n, 41092 Sevilla, Spain
e-mail: aollero@us.es

A. Torres-González
e-mail: arturotorres@us.es

J.R. Martínez-de Dios
e-mail: jdedios@us.es

This chapter deals with RO-SLAM, which only uses range measurements between the robot and static beacons deployed in the environment. Wireless Sensor Networks (WSN) nodes can be used as beacons/landmarks for RO-SLAM. In fact several SLAM methods have been proposed using WSN nodes as beacons [5, 16, 17, 19]. However, most of them use only direct measurements between the robot and the beacons, disregarding the sensing, computing and communication capabilities that WSN nodes actually have. From now on, in the chapter, the terms beacons and nodes will be used synonymously.

The integration in SLAM of inter-beacon measurements involves a number of advantages. However, despite the potential advantages, very few RO-SLAM methods that integrate inter-beacon measurements have been reported. Besides, the few that integrate inter-beacon measurements only consider single-hop readings between one robot-neigh beacon and its neighbors.

This chapter proposes an Extended Kalman Filter SLAM scheme with auxiliary Particle Filters (PFs) for delayed beacon initialization (PF-EKF SLAM) scheme that exploits inter-beacon range measurements and analyses the impact of integrating these measurements in SLAM. It presents a SLAM scheme in which the robot triggers measurement collection events in which static beacons measure the range to other beacons and re-transmit the measurements using a controlled flooding protocol. The protocol performs inter-beacon measurement collection using a configurable number of hops and naturally avoiding repeated measurements. Thus, the robot can integrate inter-beacon measurements of beacons that are very distant to the robot, drastically improving the convergence of Particle Filters (PFs). It is shown that increasing the hop number results in a drastic improvement in map estimation, which indirectly improves robot estimation too. The chapter also analyses the SLAM performance with different measurement and odometry noise levels. Although the experiments are performed using PF-EKF SLAM, the conclusions are general and can be extrapolated to any SLAM filter. Additionally, the developed scheme has been validated in real experiments in the *CONET Integrated Testbed* [12].

Thus, the main contributions of this chapter are:

- A PF-EKF SLAM scheme that integrates inter-beacon range measurements. This scheme overtakes traditional ones without inter-beacon measurements. It has lower beacon and robot estimation errors and very lower beacon initialization times and better initialization accuracy. It uses a protocol that naturally avoids flooding cycles and repeated measurements. It takes measurements from beacons that can be beyond the robot's sensing range, with a configurable number of hops.
- An analysis of the impact of integrating multi-hop inter-beacon measurements in RO-SLAM. It is obvious that integrating more and different measurements will improve the estimation, but it should be interesting to know the size of this improvement and its robustness under different conditions.
- Real experiments for validation of the proposed scheme.

- This chapter is a first step to develop scalable multi-hop measurements SLAM schemes. In this chapter we are interested in performance analysis. Scalability will be addressed by modules that dynamically adapt the number of hops in measurement. The development and validation of these modules is under current research.

The chapter is organized as follows. Section 1.1 presents the motivation of this work. Section 2 briefly summarizes the main existing RO-SLAM methods. Section 3 describes a basic RO-SLAM method, the PF-EKF without any inter-beacon measurement. Section 4 describes the collection protocol and the integration of multi-hop inter-beacon measurements in SLAM. Simulation results and evaluation of the proposed scheme under different odometry and measurement noise levels are in Sect. 5. The proposed scheme is validated in real experiments in Sect. 6. Conclusions and future works are in Sect. 7.

1.1 Motivation

Our work is motivated by schemes of cooperation between robots and sensor networks. Consider a GPS-denied environment, where a large number of sensor nodes have been deployed at unknown locations. For instance, they have been randomly thrown for monitoring a disaster or an accident in an industrial or urban area, where preexisting infrastructure can be damaged. The basic role of each sensor node is to periodically take measurements (e.g., toxic gas concentration), filter and compute statistics of the measurements and transmit them to a base station. For these tasks, the nodes must be endowed with sensing, communication and computational capabilities. In fact, this is the case of most commercial off-the-shelf (COTS) sensor nodes. Assume that each node is equipped with a range sensor and can measure the distance to the robot or to other nodes within its sensing zone. Again, this is not a constraint. For instance, most COTS nodes can measure the received signal strength (RSS) from incoming messages and can measure the range to the emitting node [1].

The application is to monitor the status of the event at the base station [6]. Having accurate estimations of the locations of the robot and of each node is necessary for accurate event monitoring and also enables advanced robot-sensor network cooperation strategies of interest in these problems, such as using the robot for sensor node deployment [13], replacement [7, 14] or collecting data from the sensors [10], among others. GPS cannot be used. A RO-SLAM method using sensor nodes (beacons) as landmarks can be very useful for on-line estimating of node and robot locations. The robot's initial pose can be used to relate this local map to global coordinates. RO-SLAM can have advantages w.r.t. visual SLAM in this problem, where suitable lighting conditions are not granted. Besides, using beacons as landmarks naturally solves data association.

Using sensor network nodes as landmarks can be very useful in SLAM. Nodes can measure the range to each other and collect the measurements using ad hoc protocols. Thus, we can assume that beacons can organize themselves into networks.

Then, each beacon can opportunistically measure its range to the robot or to other beacons, buffer and process the measurements and transmit them to other beacons or to the robot. In our approach the robot triggers a controlled flooding protocol in which beacons measure ranges to other beacons and retransmit them. The number of hops in the flooding can be configured and more and further measurements will be taken and collected.

2 Related Work

RO-SLAM relies only on range measurements, which inherently cause the problem of partial observability: only one measurement is insufficient to constrain one location. Thus, RO-SLAM methods require the robot to move and integrate measurements from different positions in order to initialize the landmark locations. Two basic approaches have been used to solve landmark initialization: directly introducing the measurements using a multi-hypothesis SLAM filter (undelayed), or combining the SLAM filter with tools for initializing landmarks (delayed). Examples of tools for delayed initialization are trilateration, probability grids and particle filters (PFs).

Trilateration methods, although simple and efficient, are very (too) sensitive to measurement noise. It was the first approach, but was soon discarded, since the SLAM performance is very dependent on the accuracy in beacon initialization. Probability grids provide better initialization, but their accuracy depends on the size and resolution of the grid. Particle filters (PFs) are maybe the most widely used landmark initialization tools in RO-SLAM. They provide better accuracy and a good number of mechanisms have been developed to reduce their computational burden.

Extended Kalman Filter (EKF) is probably the most commonly adopted approach in SLAM. EKFs combined with probability grids have been proposed in [18] and with trilateration in [16]. Particle filters use a set of samples to approximate any probability distribution. PF-EKF SLAM methods have been proposed in works such as [15]. PFs have also been combined with Rao-Blackwellised Particle Filters (RBPF) in works such as [3, 11]. Undelayed SLAM schemes address the multi-hypothesis problem without requiring specific initialization tools. In [4] EKF was combined with methods based on Sum of Gaussians.

All the aforementioned methods considered only direct robot-beacon range measurements. Despite its potential advantages, the integration of inter-beacon measurements in RO-SLAM has been very scarcely researched. The general idea of using inter-beacon measurements was given in [8], in which different ways for integrating inter-beacon measurements were proposed, using virtual nodes and adopting off-line map improvement using multidimensional scaling (MDS). MDS with inter-beacon measurements was also used in [2]. These off-line approaches are not suitable for most applications, which require on-line map and robot locations. In previous works we have integrated inter-beacon in SLAM using Sparse Extended Information Filters [21] and Sums of Gaussians [22]. However, all the research focused on integrating single-hop inter-beacon measurements.

This chapter studies the impact of multi-hop inter-beacon measurements in RO-SLAM analyzing its performance and the impact of noise on its performance. It proposes a SLAM scheme that uses a protocol to collect inter-beacon measurements using controlled flooding with a configurable number of hops. With high number of hops it can collect measurements between two beacons that are further beyond the robot's sensing range. Increasing the hop number increases the speed and accuracy of landmark initialization, which indirectly improves robot estimation.

3 PF-EKF Range Only SLAM

Assume an environment in which a number of beacons have been deployed at unknown static locations. Each beacon i can measure its distance to every beacon j within its sensing range. One robot can collect robot-beacon range measurements and integrate them in a SLAM filter.

This section briefly summarizes a RO-SLAM method with direct robot-beacon measurements as an introduction to the proposed RO-SLAM with interbeacon measurements, which is presented in Sect. 4.

3.1 EKF Range Only SLAM

EKF-SLAM is a well-known technique that recursively solves the online SLAM problem where the map is feature-based. It estimates at the same time the robot pose and the position of the features-beacons in our problem. Thus, the state vector adopted is:

$$\mathbf{x}_k = [x_k, y_k, \theta_k, x_{1,k}, y_{1,k}, \dots, x_{n,k}, y_{n,k}]^T, \quad (1)$$

where $[x_k, y_k, \theta_k]^T$ is the robot position and heading at time k and $[x_{i,k}, y_{i,k}]^T$ is the location of static beacon i .

During the EKF-SLAM prediction phase the mean μ_k and covariance Σ_k of the state \mathbf{x}_k are updated as follows:

$$\mu_k^- = f(\mu_{k-1}, u_k), \quad (2)$$

$$\Sigma_k^- = A_k \Sigma_{k-1} A_k^T + Q_k, \quad (3)$$

where Q_k is the covariance matrix that models the uncertainty in $f(\mu_{k-1}, u_k)$, the robot state transition model. A_k is the Jacobian of f :

$$A_k = \left. \frac{\partial f}{\partial \mathbf{x}_k} \right|_{\mathbf{x}_k = \mu_{k-1}} \quad (4)$$

When the robot receives a new range measurement $z_{i,k}$ from beacon i , the update phase of the EKF can be performed using the observation model h_i that relates \mathbf{x}_k and $z_{i,k}$. In our case, h_i models the distance between the robot and beacon i :

$$z_{i,k} = h_i(\mu_k) = \sqrt{(x_k - x_{i,k})^2 + (y_k - y_{i,k})^2} \quad (5)$$

$R_{i,k}$ is the variance of the uncertainty in $h_i(\mu_k)$. Hence, the mean and covariance of the state can be updated as:

$$\mu_k = \mu_k^- + K_k(z_k - h(\mu_k^-, i)), \quad (6)$$

$$\Sigma_k = (I - K_k H_k) \Sigma_k^-, \quad (7)$$

where H_k is the Jacobian of h_i and K_k is the Kalman gain. For a more detailed description of the EKF algorithm refer to [20].

3.2 PF Initialization

Range Only SLAM deals with partial observation. In this chapter we use an auxiliary Particle Filter (PF) for beacon initialization. PFs can represent any probability distribution and can naturally solve the partial observability problem due to their multi-hypothesis capability. When the robot receives the first range measurement $z_{i,k}$ from beacon i , it initializes an auxiliary PF for beacon i , PF_i , in which the particles, which are hypotheses of the beacon location, are spread around the robot at distances drawn from an annular distribution with mean $z_{i,k}$ and width that depends on the measurements variance.

When the robot receives a new measurement from beacon i , the particles of PF_i are updated making them to condensate towards the real beacon position. PFs implement the so-called *importance resampling*, which draws with replacement of M particles. The probability of drawing each particle is given by its importance weight. *Importance resampling* helps to have a faster convergence of the PF. In the experiments shown in this chapter we replaced $M=10\%$ of the particles. We considered that a PF has converged in a Gaussian distribution when the covariance of its particles is lower than a certain value.

When PF_i converges, the beacon estimated location $[x_i, y_i]^T$ is computed as the weighted mean of all particles and it is added to the EKF state vector q_k . The flexibility of PFs is at the expense of increasing computer burden. PF iterations require significantly more computer burden than EKF iterations. That is why PF convergence is widely taken as a metric for computer cost.

Along the PF-EKF SLAM iterations each static beacon can be in two stages: the PF stage and the EKF stage. When the robot receives a measurement from a beacon, it will be used to update its PF or the EKF depending on the beacon stage.

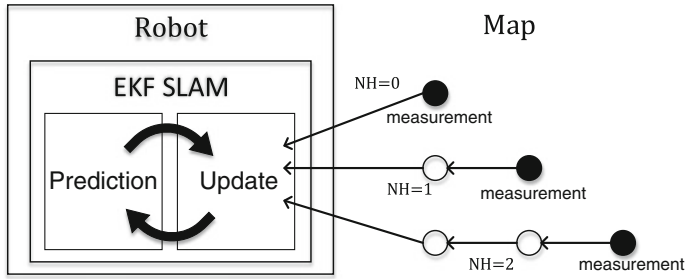


Fig. 1 Operation of the proposed scheme. *Black circles* represent the beacons which measurements are taken from. *White circles* represent beacons that retransmit the measurements back to the robot

4 Integration of Inter-beacon Measurements

The integration of inter-beacon measurements can involve a number of advantages. The robot can anticipate by integrating measurements between beacons beyond its sensing range, resulting in sooner and more accurate beacon initialization. Also, inter-beacon measurements are useful to improve the accuracy of the map estimation. Figure 1 shows a block diagram summarizing the operation of the proposed scheme.

This section extends the SLAM method described in Sect. 3 with inter-beacon measurements. It is divided in the following parts:

- Section 4.1 describes the inter-beacon measurements collection protocol.
- Section 4.2 presents the extended RO-SLAM method. It details how inter-beacon measurements are integrated in the adapted SLAM filter.

4.1 Inter-beacon Measurements Collection Protocol

Each measurement collection event is divided in two parts: the forward stage and the backward stage. The forward stage performs a cascade-like measuring of inter-beacon distances. The origin of the cascade is the robot and its depth is NH , the hop number. All messages in this stage include a field nh representing the number of hops remaining until the end of the forward stage. The backward stage orderly collects measurements from the involved beacons using backward messages. All messages include a sequence number Seq that identifies the measurement event. Each beacon tracks the Seq of the last measurement event it was involved in.

The forward stage starts when the robot broadcasts a forward message with $nh = NH$ to the beacons within its sensing range. Each beacon i receiving the message checks if it is a new measurement event. If it is not, the message is ignored. Otherwise, beacon i updates $nh = nh - 1$, measures its distance to all the beacons in its sensing range and buffers the measurements in ms_i , the measurement set for beacon i . If $nh > 0$, it broadcasts a forward message with the new nh . The beacon keeps the

ID of its parent beacon—from which it received the forward message—and starts its backward stage. If $nh = 0$, that forward message reached its hop limit and it is not retransmitted: the forward stage ends and the backward stage starts. Then, beacon i creates a backward message with ms_i and sends it to its father. In the backward stage each beacon updates its measurement set. When beacon i receives a backward message with a suitable Seq , it adds to ms_i the measurements contained in the message. Each beacon keeps in backward stage until its timeout expires. Then, it sends a backward message containing ms_i to its parent beacon.

Figure 2 illustrates its operation with different NH . With $NH = 0$ —the traditional approach—the robot does not broadcast the forward message and only collects robot-beacon measurements $\{z_{r,1}, z_{r,2}\}$. With $NH = 1$, the robot collects $\{z_{r,1}, z_{r,2}, z_{1,3}, z_{1,4}, z_{1,r}, z_{2,r}\}$. With $NH = 2$, among others the robot collects measurements between beacons that are beyond its sensing range such as $z_{3,5}$. The proposed protocol can dynamically change NH . Also, notice that it prevents flooding cycles, naturally avoiding repeated measurements and canceling the need for additional filtering.

With $NH = 1$, the robot broadcasts a forward message with $nh = 1$. Beacon $ID = 1$ receives the message, updates $nh = 0$ and measures distances $z_{1,j} \forall j \in SZ_1$, being SZ_1 the sensing zone of beacon $ID = 1$. Since $nh = 0$, it sends a backward message with ims_1 to the robot—its father. At the end of the backward stage the robot has collected the following measurements $\{z_{r,1}, z_{r,6}, z_{1,2}, z_{1,3}, z_{1,r}, z_{6,r}\}$. The protocol obtains two measurements for each inter-beacon distance except for the deepest beacons, of which one is collected. Figure 2-right shows its operation with $NH = 2$ and the measurements collected by each beacon and the robot: it collects measurements between beacons that are beyond its sensing range. In this example beacon $ID = 1$ received three forward messages. The first one was sent by the robot. It also received one from beacon $ID = 2$ and one from $ID = 3$ but they were ignored since they had non-new Seq values.

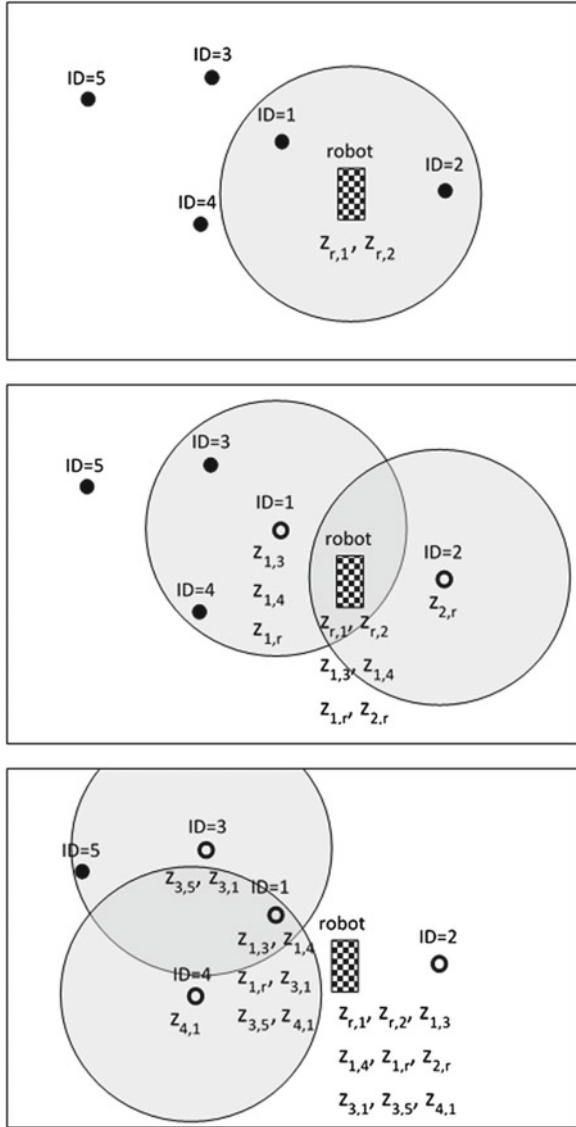
4.2 RO-SLAM Extended with Inter-beacon Measurements

This section describes how inter-beacon measurements are integrated in SLAM. With $NH = 0$, the proposed measurement collection protocol takes only robot-beacon measurements: they are integrated as in Sect. 3. In general with $NH > 0$ the proposed protocol takes measurements between two beacons, being at least one of them at NH or less hops from the robot. The observation model for inter-beacon measurement $z_{i,j}$ between beacons i and j used is similar to that in (5) but adapted to consider the range between the estimated locations of both beacons. $z_{i,j}$ is integrated in SLAM differently depending on the stage of the involved beacons.

If both beacons are in the EKF stage, $z_{i,j}$ is used to update the EKF using the following observation Jacobian:

$$H_{i,j} = \left[\dots, 0, \frac{x_{i,k} - x_{j,k}}{h_{i,j}}, \frac{y_{i,k} - y_{j,k}}{h_{i,j}}, 0, \dots, 0, \frac{x_{j,k} - x_{i,k}}{h_{i,j}}, \frac{y_{j,k} - y_{i,k}}{h_{i,j}}, 0, \dots \right] \quad (8)$$

Fig. 2 Examples of measurements collection with: $NH = 0$ (top), $NH = 1$ (center) and $NH = 2$ (bottom). Grey circles represent the sensing zones



All the terms in $H_{i,j}$ are zero except those for the entries corresponding to the locations of beacons i and j .

Figure 3 summarizes the operation of the algorithm when a new inter-beacon measurement is taken. If only one of the beacons, e.g. beacon j , is in the EKF stage, $z_{i,j}$ is used to update or initialize the PF of beacon i . If neither beacon i nor beacon j are in the EKF stage, the measurement is kept for future use until one of the PFs, either PF_i or PF_j , converges. Instead of buffering all measurements, for simplification and

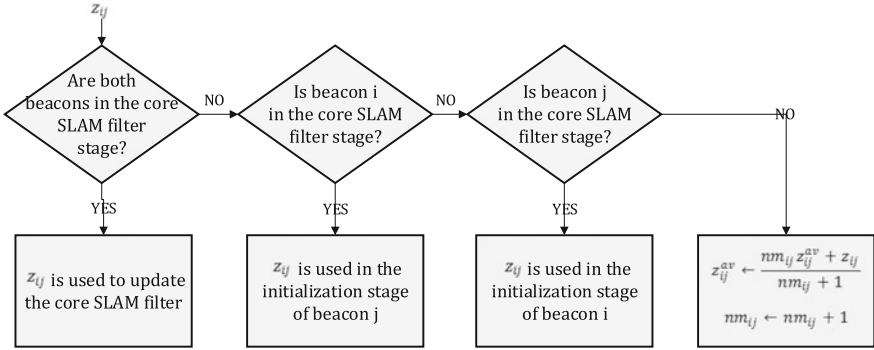


Fig. 3 Integration of inter-beacon measurements in the SLAM filter

efficiency, the robot only keeps the number— $nm_{i,j}$ —and mean of the measurements. Assuming Gaussian noise, the average value is considered as a measurement but with variance $nm_{i,j}$ times lower than that of the range measurements.

As a result, in the proposed scheme the convergence of a PF triggers the integration of low-variance inter-beacon measurements, helping the convergence of other PFs and enabling a chain-reaction PF convergence effect. This effect drastically reduces the PF convergence times and helps to anticipate the deployment of other PFs. The effect is larger with higher numbers of inter-beacon relations, i.e. the number of beacons of which measurements can be taken by each beacon. Higher NH also enables the robot to collect measurements from a higher number of inter-beacon relations.

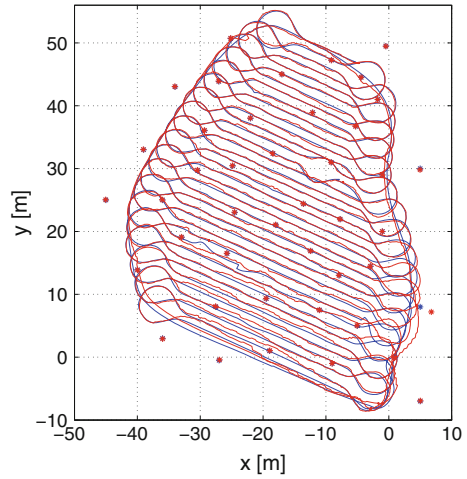
5 Simulations

This section is divided in two parts. The first evaluates the effects of using different values of NH while the second one analyzes its performance with different odometry and measurement error levels. For comparison purposes we used the robot ground truth and odometry from the Djugash Plaza dataset [9]. However, we assumed a larger number of static beacons—43—scattered in the scenario. We have repeated these simulations with other number of beacons and obtained similar qualitative and quantitative conclusions. The robot kinematic model used was the following:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + T_k V_k \sin \theta_{k-1} \\ y_{k-1} + T_k V_k \cos \theta_{k-1} \\ \theta_{k-1} + T_k \alpha_k \end{bmatrix}, \quad (9)$$

where (x_k, y_k, θ_k) is the robot state, V_k and α_k are respectively the odometry linear and the steering velocities and T_k is the differential time between t_k and t_{k-1} .

Fig. 4 Results of the proposed scheme with $NH = 2$



Each beacon is assumed to be equipped with a range sensor with 10m sensing range and a standard deviation $\sigma_m = 1$ m. PFs are initialized by deploying 300 particles in an annular distribution with radius $z_{i,k}$ and width $3\sigma_m$. SLAM provides the estimations in a local coordinate frame. To compare with the ground-truth, an affine transform is performed to the solution given by SLAM, re-aligning the local solution into the same global coordinate frame.

Figure 4 shows the result of the proposed scheme with $NH = 2$. The robot and static beacons estimated locations are represented in red color while the ground truth is in blue. In order to assess the performance of the proposed approach, series of 300 simulations were performed with $NH = 0$, $NH = 1$ and $NH = 2$. Every simulation consider random beacon settings and robot trajectories. Figure 5 shows the mean

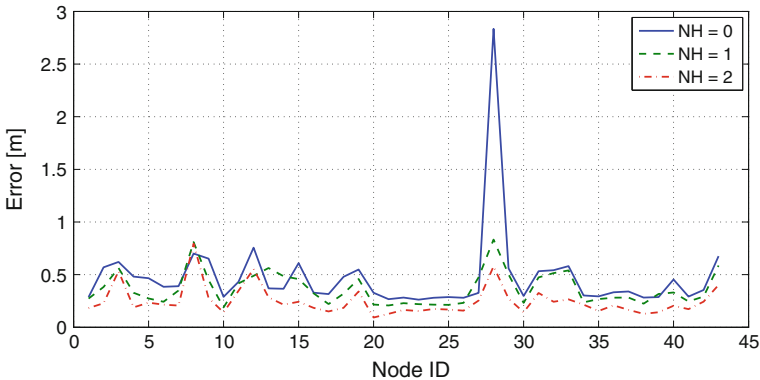
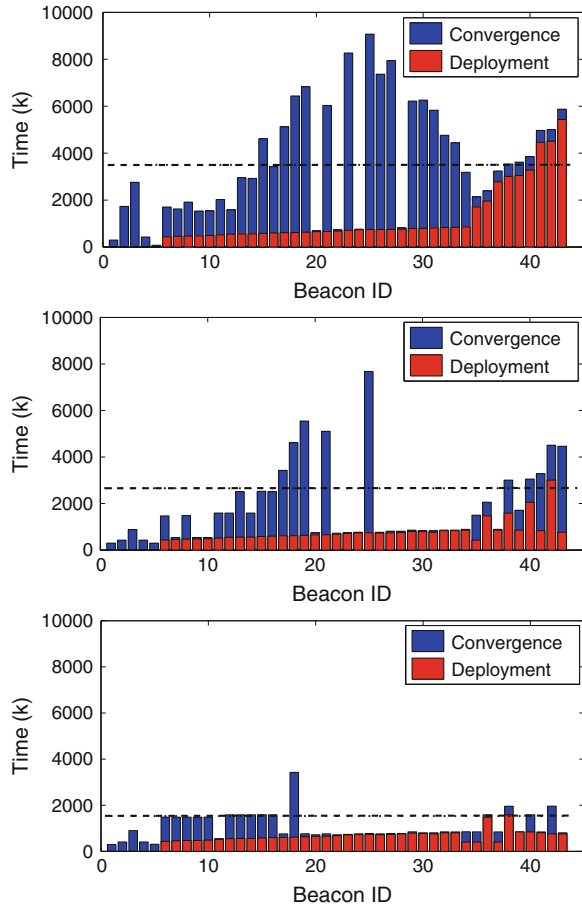


Fig. 5 Mean error in the location of each static beacon with different NH

Fig. 6 Deployment (red) and convergence (blue) times of each beacon PF with $NH = 0$ (top), $NH = 1$ (center) and $NH = 2$ (bottom). Black dashed lines represent the average convergence time for each value of NH



error in the location of each static beacon in the three cases. The map error reduced significantly with higher NH .

Figure 6 shows the mean time when the PF of each beacon were deployed (red color) and converged (blue). PFs converged in average at time $k = 3503$ with $NH = 0$, at $k = 2657$ with $NH = 1$ and at $k = 1532$ with $NH = 2$. The PF convergence chain reaction effect is emphasized with higher NH . The improvement can be noticed in almost all beacons and is particularly evident in beacons that are distant from the robot initial position. With $NH = 2$ the multi-hop flooding protocol allows the robot to integrate measurements between two beacons beyond the robot's sensing range. Thus, some PFs converge even before the robot takes a first direct measurement for that beacon and these robot-beacon measurements are used directly in the EKF improving the beacon and robot estimations. This improvement can also be noticed in the PF deployment times. The average deployment time for $NH = 0$ was time $k = 1142$, while it was $k = 861$ for $NH = 1$ and $k = 609$ for $NH = 2$.

Table 1 Performance evaluation of the proposed scheme with $NH = 1$ w.r.t. the traditional approach, $NH = 0$

	$\sigma_{m,1}$ (%)	$\sigma_{m,2}$ (%)	$\sigma_{m,3}$ (%)	$\sigma_{m,4}$ (%)	
Mean map error	33.5	32.9	30.2	27.3	$\sigma_{o,1}$
Mean robot error	9.7	9.2	8.6	7.2	
Mean init. time	60.1	59.9	56.8	55.9	
Mean map error	41.7	40.3	38.4	34.1	$\sigma_{o,2}$
Mean robot error	11.2	10.8	10.3	9.4	
Mean init. time	60.5	59.7	57.8	56.2	
Mean map error	45.6	43.8	42.2	37.9	$\sigma_{o,3}$
Mean robot error	12.4	11.9	11.1	10.0	
Mean init. time	60.5	60.0	57.1	56.0	

The overall number of measurements integrated in each simulation were: $\overline{nm} = 10046$ for $NH = 0$, $\overline{nm} = 22378$ for $NH = 1$ and $\overline{nm} = 31958$ for $NH = 2$. In order to have an estimate of the resource consumption, the computing times until 90% of the PFs had converged have been calculated. They were $\bar{t} = 9.32$ s ($NH = 0$), $\bar{t} = 8.38$ s ($NH = 1$) and $\bar{t} = 8.01$ s ($NH = 2$). These computing times have been calculated through *Matlab Profiler*, running in a i7-3630QM computer. The large increase in the number of measurements with higher NH was compensated with shorter PF convergence times, resulting in overall computational burden savings.

Tables 1 and 2 compare the robustness of using different NH values assuming three odometry error levels—good ($\sigma_{o,1} = 0.05$ m/s), average ($\sigma_{o,2} = 0.15$ m/s) and bad ($\sigma_{o,3} = 0.25$ m/s)—and four measurement inaccuracies—good ($\sigma_{m,1} = 0.1$ m), average ($\sigma_{m,2} = 0.5$ m), bad ($\sigma_{m,3} = 1$ m) and very bad ($\sigma_{m,4} = 1.5$ m). The tables summarize the improvement in performance originated from using inter-beacon measurements with $NH = 1$ and $NH = 2$ w.r.t. the traditional approach— $NH = 0$. They show the mean results from 200 simulations with different robot paths and randomly deployed beacons.

It is shown that SLAM performance and robustness to noise level improves when increasing the hop number of inter-beacon measurements: the performance increment w.r.t. $NH = 0$ is higher with $NH = 2$ than with $NH = 1$. The use of inter-beacon measurements reduce the PF convergence times approximately in the same way (about 60% with $NH = 1$ and 70% with $NH = 2$) despite the odometry and measurement error levels. Also, the improvement in map estimation is more evident

Table 2 Performance evaluation of the proposed scheme with $NH = 2$ w.r.t. the traditional approach, $NH = 0$

	$\sigma_{m,1}$ (%)	$\sigma_{m,2}$ (%)	$\sigma_{m,3}$ (%)	$\sigma_{m,4}$ (%)	
Mean map error	48.2	46.8	43.5	39.7	$\sigma_{o,1}$
Mean robot error	17.6	16.9	16.1	14.9	
Mean init. time	72.3	71.2	70.4	66.7	
Mean map error	54.8	52.9	50.1	46.3	$\sigma_{o,2}$
Mean robot error	19.5	19.0	18.1	16.8	
Mean init. time	72.6	71.2	70.6	66.9	
Mean map error	65.1	63.5	61.2	57.4	$\sigma_{o,3}$
Mean robot error	20.7	20.1	19.3	18.2	
Mean init. time	72.7	71.4	70.9	67.2	

than in robot localization. This is attributed to the fact that inter-beacon measurements directly update landmark estimations and influence indirectly on the robot estimation.

It can also be noticed that the use of multi-hop inter-beacon measurements provides higher improvements w.r.t. $NH = 0$ with lower measurement noise levels and worse odometry. In this case the integration of inter-beacon measurements highly improves the estimation. With higher measurement noise levels and better odometry, inter-beacon measurements are less useful and cause lower improvements. On the other hand, inter-beacon measurements increase the coupling in the SLAM filter. This is a significant advantage but could also influence divergence in extreme cases with very high measurement noise. We noticed these divergence effect with measurements with $\sigma_m > 2$ m, which has low practical implications since most COTS range sensors used in SLAM usually have better accuracies.

6 Experiments

The proposed scheme has been validated in real experiments performed in the *CONET Integrated Testbed* [12], see Fig. 7-left. The testbed is a remote open tool to assess and compare methods and algorithms combining multi-robot systems and

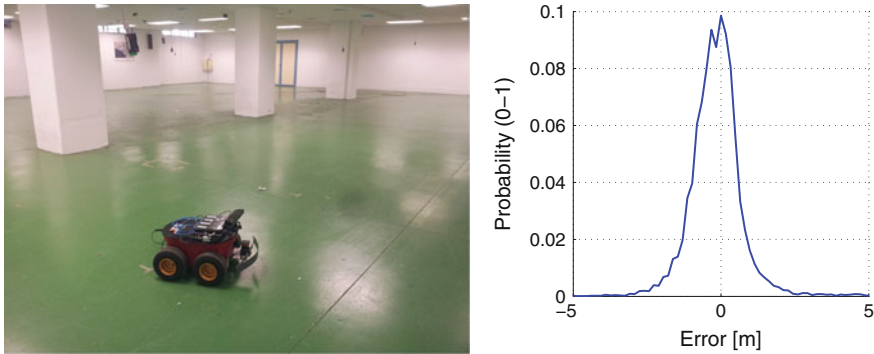


Fig. 7 (Left) Picture taken in the validation experiments. (Right) Characterization of localization errors of *Nanotron NanoPAN* nodes

ubiquitous networks. The testbed is composed by 5 *Pioneer 3-AT*¹ robots, each equipped with a *Hokuyo UTM-30LX*² range finder and a *Microsoft Kinect* camera, GPS and Inertial Measurement Units, among others. The testbed also includes a network of static cameras, WSN nodes and a network of 6 *Nanotron nanoPAN 5375*³ beacons. *Nanotron beacons take range measurements using TDOA (Time Difference of Arrival). The experiments were performed using one of the Pioneer 3-AT robots equipped with one of the Nanotron nodes. The other 5 were deployed in the testbed hanging from the ceiling, at 1.2m height. Although, the system is only tested in 2D, and the difference in height with the robot has been treated adding this difference to the observation model (5). We performed experiments to characterize the error of Nanotron for indoors. They had an error of zero mean and a variance $\sigma_m^2 = 1$ m, see Fig. 7-right.*

We performed series of experiments with $NH = 0$ and $NH = 1$, Fig. 8. $NH = 2$ was unnecessary in this scenario due to the large sensing range of the beacons used and the low number of beacons. Map and robot localization errors were both lower with $NH = 1$. Figure 9 shows the evolution of the location error for each of the 5 static beacons with $NH = 0$ and $NH = 1$. The drawing for each beacon starts when its auxiliary PF converged. Beacon 1 was the first to converge in both cases. It converged at same times with $NH = 0$ and $NH = 1$. However, with $NH = 1$ the rest of the beacons converged shortly after due to the aforementioned chain reaction effect. With $NH = 0$ they required significantly longer times.

Also, with $NH = 1$ PFs converged with lower error than with $NH = 0$: $\bar{e} = 0.35$ m for $NH = 1$ while $\bar{e} = 0.47$ m for $NH = 0$. Along the experiment, beacons location errors were also lower with $NH = 1$ than with $NH = 0$. Consequently, the robot localization accuracy was also better, see Table 3. The experimental setting has been simulated obtaining similar results. The difference between the performance

¹<http://www.mobilerobots.com/ResearchRobots/P3AT.aspx>.

²https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html.

³http://www.nanotron.com/EN/PR_ic_modules.php.

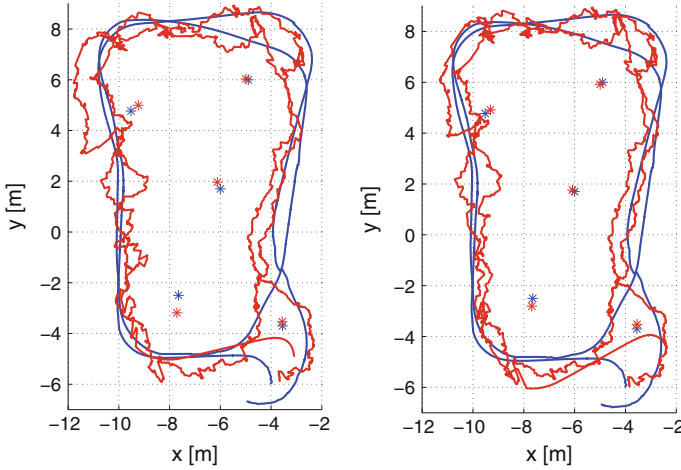


Fig. 8 Results of the method with $NH = 0$ (left) and $NH = 1$ (right). In both cases an affine transform was performed on the final robot’s path and beacon estimations, re-aligning the local solution into the same global coordinate frame. *Blue lines and markers* represent ground-truth positions of the robot and the map while *red* is used for estimations of the proposed scheme

Fig. 9 Evolution of the location error for each static beacon with $NH = 0$ (top) and $NH = 1$ (bottom)

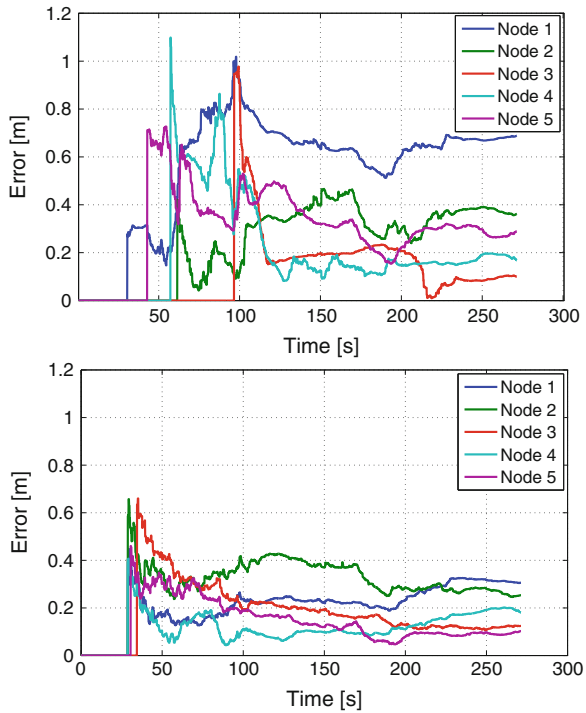


Table 3 Results of the experiments

	$NH = 0$	$NH = 1$
Mean map error	0.33 m	0.19 m
Mean robot error	0.72 m	0.79 m
Mean init. time	57.2 s	30 s

of the real experiment and simulation is lower than 5% both in map and robot location along the experiments. We attribute this likelihood to the care taken during the modeling both of the robot kinematic model and odometry noise levels and also to the observation model and uncertainty characterization of the *Nanotron* devices. Thus, the above confirm the simulations and validate the proposed scheme.

Scalability is a critical issue in SLAM and in WSN. Unfortunately, the analysis of the scalability is out of the scope of this chapter but this issue is clearly in our mind. We noticed that having $NH = 2$ is particularly useful during map initialization. Of course, using $NH = 2$ involves high resource consumption and can have scalability issues. However, after the map has been initialized, the increment in resource consumption involved by $NH = 2$ is not balanced with its increment in accuracy. Thus, it is recommendable to stop $NH = 2$ and use $NH = 0$ instead, originating a solution in which the map converges as fast as with $NH = 2$ but is as scalable as standard SLAM schemes (with $NH = 0$). We are analyzing and working in different schemes of dynamically modifying the value of NH depending on the environment and status of the experiment. From this point of view, this chapter is a starting point in this new research line.

7 Conclusions

This chapter analyzes the impact on SLAM of integrating inter-beacon measurements with different hop number. It presents a SLAM scheme that includes a protocol that collects—naturally avoiding repetitions—measurements between static beacons using a configurable number of hops. Thus, the SLAM filter integrates measurements between beacons that can be far beyond the robot’s sensing range. In this scheme the convergence of the first PF triggers a PF convergence chain reaction, drastically speeding up PF convergence and anticipating PF deployment, significantly improving map estimations. This effect is larger when integrating measurements with higher hop number.

The chapter also analyzes the robustness to measurement and odometry noise levels of the scheme when integrating measurements with different hop numbers. Its advantages w.r.t. traditional schemes are more evident with lower measurements error levels and higher odometry error. Also, its performance improves as the hop number is increased. To the best of our knowledge this chapter is the first that uses multi-hop

inter-beacon measurements in SLAM. It opens wide fields for future research. In this chapter the scheme has been validated through simulations and real hardware experimentation in the *CONET Integrated Testbed* (<https://conet.us.es>) [12].

The extension of this scheme to three dimension environments is object of current development. The application in flying systems of more accurate SLAM schemes could be interesting. This chapter is a first step to develop scalable multi-hop measurements SLAM schemes. Thus, future works should also analyze and develop dynamic schemes for selecting the number of hops and the inter-beacon measurement collection frequency.

References

1. Banatre, M., Marron, P., Ollero, A., Wolisz, A.: *Cooperating Embedded Systems and Wireless Sensor Networks*. Wiley, London (2008)
2. Bardella, A., Danieletto, M., Menegatti, E., Zanella, A., Pretto, A., Zanuttigh, P.: Autonomous robot exploration in smart environments exploiting wireless sensors and visual features. *Annals of telecommunications-Annales des télécommunications* **67**(7–8), 297–311 (2012)
3. Blanco, J.L., Fernandez-Madrigal, J.A., Gonzalez, J.: Efficient probabilistic range-only slam. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'08*, pp. 1017–1022 (Sept 2008)
4. Caballero, F., Merino, L., Ollero, A.: A general gaussian-mixture approach for range-only mapping using multiple hypotheses. In: *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'10*, pp. 4404–4409 (May 2010)
5. Challa, S., Leipold, F., Deshpande, S., Liu, M.: Simultaneous localization and mapping in wireless sensor networks. In: *Intelligent Sensors, Sensor Networks and Information Processing Conference*, pp. 81–87 (Dec 2005)
6. Clemente, A.D.S.B., Martínez-de Dios, J.R., Baturone, A.O.: A wsn-based tool for urban and industrial fire-fighting. *Sensors* **12**(11), 15009–15035 (2012)
7. Martínez-de Dios, J., Lferd, K., de San Bernabé, A., Núñez, G., Torres-González, A., Ollero, A.: Cooperation between uas and wireless sensor networks for efficient data collection in large environments. *J. Int. Robot. Syst.* **70**(1–4), 491–508 (2013)
8. Djugash, J., Singh, S., Kantor, G., Zhang, W.: Range-only slam for robots operating cooperatively with sensor networks. In: *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'06*, pp. 2078–2084 (May 2006)
9. Djugash, J., Hamner, B., Roth, S.: Navigating with ranging radios: five data sets with ground truth. *J. Field Robot.* **26**(9), 689–695 (2009). Sept
10. Goerner, J., Chakraborty, N., Sycara, K.: Energy efficient data collection with mobile robots in heterogeneous sensor networks. In: *IEEE International Conference on Robotics and Automation, ICRA'13* (2013)
11. Hai, D., Li, Y., Zhang, H., Li, X.: Simultaneous localization and mapping of robot in wireless sensor network. In: *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 3, pp. 173–178 (Oct 2010)
12. Jiménez-González, A., Martínez-de Dios, J.R., Ollero, A.: An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. *Sensors* **11**(12), 11516–11543 (2011)
13. Maza, I., Caballero, F., Capitan, J.: Martínez-de Dios, J., Ollero, A.: A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities. *J. Field Robot.* **28**(3), 303–328 (2011)
14. Mei, Y., Xian, C., Das, S., Hu, Y.C., Lu, Y.H.: Sensor replacement using mobile robots. *Comput. Commun.* **30**(13), 2615–2626 (Sep 2007)

15. Menegatti, E., Danieleto, M., Mina, M., Pretto, A., Bardella, A., Zanconato, S., Zanuttigh, P., Zanella, A.: Autonomous discovery, localization and recognition of smart objects through wsn and image features. *Proc. IEEE GLOBECOM* **2010**, 1653–1657 (2010)
16. Menegatti, E., Zanella, A., Zilli, S., Zorzi, F., Pagello, E.: Range-only slam with a mobile robot and a wireless sensor networks. In: *IEEE International Conference on Robotics and Automation, ICRA'09*, pp. 8–14 (2009)
17. Nogueira, M., Sousa, J., Pereira, F.: Cooperative autonomous underwater vehicle localization. In: *IEEE OCEANS—Sydney*, pp. 1–9 (2010)
18. Olson, E., Leonard, J., Teller, S.: Robust range-only beacon localization. In: *Proceedings of 2004 IEEE/OES Autonomous Underwater Vehicles*, pp. 66–75 (June 2004)
19. Sun, D., Kleiner, A., Wendt, T.: Multi-robot range-only slam by active sensor nodes for urban search and rescue. *RoboCup 2008: Robot Soccer World Cup XII. Lecture Notes in Computer Science*, vol. 5399, pp. 318–330. Springer, Berlin Heidelberg (2009)
20. Thrun, S., Burgard, W., Fox, D.: *Probabilistic Robotics*, 3rd edn. The MIT Press, Cambridge (2005)
21. Torres-González, A., Martínez-de Dios, J., Ollero, A.: Efficient robot-sensor network distributed seif range-only slam. In: *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2014*, vol. 31
22. Torres-González, A., Martínez-de Dios, J., Ollero, A.: Integrating internode measurements in sum of gaussians range only slam. In: *ROBOT2013: First Iberian Robotics Conference*, pp. 473–487. Springer International Publishing (2014)

A Distributed and Multithreaded SLAM Architecture for Robotic Clusters and Wireless Sensor Networks

David Portugal, Bruno D. Gouveia and Lino Marques

Abstract In this work, we propose an extremely efficient architecture for the Simultaneous Localization and Mapping (SLAM) problem. The architecture makes use of multithreading and workload distribution over a robotic cluster or a wireless sensor network (WSN) in order to parallelize the most widely used Rao-Blackwellized Particle Filter (RBPF) SLAM approach. We apply the method in common computers found in robots and sensor networks, and evaluate the tradeoffs in terms of efficiency, complexity, load balancing and SLAM performance. It is shown that a significant gain in efficiency can be obtained. Furthermore, the method enables us to raise the workload up to values that would not be possible in a single robot solution, thus gaining in localization precision and map accuracy. All the results are extracted from frequently used SLAM datasets available in the Robotics community and a real world testbed is described to show the potential of using the proposed philosophy.

Keywords Computation sharing · Simultaneous localization and mapping · Cooperative robotics · Distributed computing · Wireless sensor networks

This work was partially carried out in the framework of TIRAMISU project (www.fp7-tiramisu.eu). This project is funded by the European Community's Seventh Framework Program (FP7/SEC/284747).

D. Portugal · B.D. Gouveia · L. Marques (✉)
Institute of Systems and Robotics (ISR), University of Coimbra (UC),
Pólo II, 3030-290 Coimbra, Portugal
e-mail: lino@isr.uc.pt

B.D. Gouveia
e-mail: bgouveia@isr.uc.pt

D. Portugal
e-mail: davidbsp@isr.uc.pt

1 Introduction

The Simultaneous Localization and Mapping (SLAM) problem in Robotics has gained increasing attention over the years, and today several approaches exist to address the problem with different levels of success. With the growth of the Robotics field, algorithms for navigation, localization, mapping and other robotic tasks tended to increase in complexity over the years. However, robotic platforms commonly suffer from limited computational resources especially in multi-robot teams. The high processing demands and relatively cheap designs involved, interfere with the ability of each individual robot to finish the task underway, and in processors with limited computation power, running several of the aforementioned tasks simultaneously may not be possible. In the particular case of SLAM, Roboticians have increasingly proposed new methods to reduce measurement uncertainty and produce more accurate maps, as well as using efficient data structures and methods, such as those that take advantage of sparsity in the dependencies between data and state variables [1] or using pose graphs [2], to reduce memory and processing requirements. Inspired by a variety of network-based computing concepts, alternative strategies have been adopted to relieve the load involved in perception, reasoning, storage and computation in diverse robotic tasks. However, modern day computer architectures have not been fully explored in this context and the scarcity of closed SLAM solutions based on parallel computing techniques is surprising.

In this chapter, we propose a novel distributed approach over the network of robots or sensors, which also leverages from the multiple Central Processing Units (CPUs) embedded in modern day computers in order to speedup a widely used SLAM approach in Robotics—the grid based Rao-Blackwellized Particle Filter (RBPF) SLAM method, *GMapping*, proposed by Grisetti et al. [3]. We apply a distributed and multithreaded (MT) approach in a 2D SLAM application for the first time as far as our knowledge goes, in a static Wireless Sensors Network (WSN) and a solidary multi-robot system (a mobile WSN), where computing nodes occasionally share resources to solve heavy algorithm steps.

Our architecture is built upon the concept of *Robotic Cluster* proposed by Marjovi et al. [4], which is defined as a group of individual robots able to share their processing resources among the group in order to quickly solve computationally hard problems arising in the real world. Applications of *Robotic clusters* usually demand high processing power and relatively cheap designs. This cooperative mechanism can be applied using robots' wireless communication capabilities to share resources, and an evident application for the *Robotic Cluster* concept is SLAM. Generally in these scenarios, mobile robots produce large amounts of data that must be processed in real-time [5], computation peaks occur regularly, and the latency when connecting to the cluster and querying available computing nodes is negligible.

In the next section, seminal work on SLAM, and efficient parallel architectures and network-based computing is reviewed. In Sect. 3, the proposed distributed and multithreaded architecture for solidary SLAM is described in detail. Afterwards, in Sect. 4 we discuss the results of applying the approach described in this paper,

which is compared against the classical RBPF *GMapping* SLAM algorithm. We also analyze the benefits in performance of turning to distributed computing sharing over the single robot solution, and we validate the work through real world experiments on physical robots with limited processing capabilities. Finally, the article ends with conclusions and future directions of research.

2 Related Work

In the field of mobile robotics, typical solutions for the problem of Simultaneous Localization and Mapping (SLAM) rely on probabilistic frameworks, which account for sensor measurement noise and estimation uncertainty. Classical techniques make use of Kalman Filters (KFs) [6] and Particle Filters (PFs) [1] to incrementally compute joint posterior distributions over robot poses and landmarks. Along the years, several enhancements based on these approaches have been described in the literature, such as Extended Kalman Filters (EKFs) and Rao-Blackwellized Particle Filter (RBPF)s. In addition, graph-based algorithms, e.g. [2, 7], are highly in focus due to the efficiency gained when maintaining large-scale maps, which stems from discarding irrelevant measurements and graph optimization processes.

Also popular nowadays are methods which take advantage of high scanning rates of modern day Light Detection And Ranging (LIDAR) technology. These methods rely heavily on scan matching of consecutive sensor readings, with combination of other techniques like multi-resolution occupancy grid maps [8] or dynamic likelihood field models for measurement [9]. Despite the evident advancements in SLAM research, e.g. [10], a robot with such capabilities still has to be equipped with appropriate computation power to adequately handle the processing and memory requirements.

Beyond the usage of optimized data structures and algorithms, some authors have turned to modern day computer architectures to handle heavy processing SLAM in real-time. According to [11], multiprocessing is extremely efficient in terms of energetic consumption since the speed requirements on each parallel unit are reduced, allowing for a reduction in voltage.

For example, the authors in [12] propose a simple and effective scan-matching module that can be potentially used in several SLAM algorithms, which takes advantage of the Single Instruction, Multiple Data (SIMD) units available in modern processors. This allows the processor to execute an operation over a data vector in the same instruction time. Using this approach, an average speedup of 3.5 compared to the non-SIMD version of the algorithm was observed. Additionally, the authors in [13] have proposed a Visual SLAM module implemented in the Compute Unified Device Architecture (CUDA), which runs at a 15 Hz rate. Their approach performs sparse scene flow, real-time feature tracking, visual odometry, loop detection and global mapping in parallel. Similarly, Par and Tosun [14] have addressed classical PF-based Sequential Monte Carlo Localization of a vehicle in a known map with multicore and manycore processors. They have used the Open Multi-Processing

(OpenMP) programming model for the parallelization of the predict and update phases of the PF in a multicore CPU, and also implemented a Graphics Processing Unit (GPU) version, obtaining speedup values of up to 4.7 and 75 respectively. In [15], a parallel implementation of a RBPF was also addressed. The authors not only propose to assign different particles among available processing elements in order to compute their weights, but also to conduct the resampling process. Despite the potential of the work, it was only validated by tracking a maneuvering vehicle in Matlab simulations.

Related philosophies for distributing the computation load in robotic application are emerging. This is the case of Cloud Robotics. In [16], robots performing visual SLAM query a Cloud service to run demanding steps of the algorithm and allocate storage space, thus freeing the robot embedded computers from most of the computation effort. This work is part of the RoboEarth project, which provides a Cloud Robotics infrastructure, more specifically a giant network and database repository for sharing information and triggering learning behaviors for robots.

In [17], a cloud-inspired framework named Distributed Agents with Collective Intelligence (DAvinCi) is presented. The authors use a powerful Hadoop¹ system composed of eight Intel quad core server nodes, which works as a private cloud computing environment for service robots in large environments. A Rao-Blackwellized Particle Filter RBPF grid-based Simultaneous Localization and Mapping (SLAM) implementation was used as demonstration. The robots are basically mobile sensing nodes, which send data to the DAvinCi server that works as a proxy to the Hadoop framework, which takes care of all the computation. With this powerful parallel computing system, the authors were able to obtain a speedup ratio close to 7.3. However, system scalability assessment was not conducted.

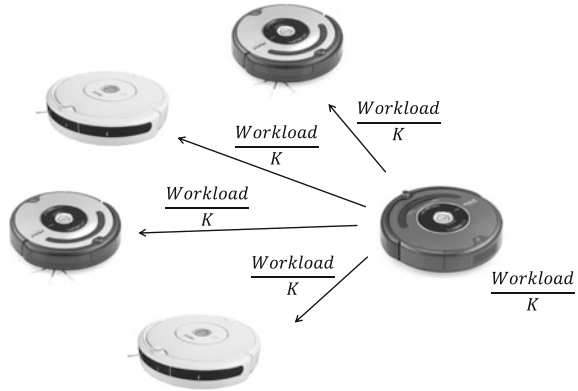
The two previously described works imply the continued availability of the Cloud entity. This may not be possible in tasks involving distributed multi-robot systems over wide areas or outdoor environments. In addition, the connection to/from the Cloud represents a critical point of failure in the system, and the available bandwidth may become a bottleneck on the communication flow in systems with high number of agents.

In the last decades an evident effort has been made in the area of networked robots ever since the pioneering work described by Siegwart et al. [18] on teams of mobile robot systems connected to the web, and more recently multiprocessor architectures have been emerging in the context of multi-robot systems, e.g. in the scope of the Swarmanoid project [19], where CPU-intensive tasks are allocated to the main processor of each robot and real-time sensor readings are allocated to several microcontrollers. This represented an important architectural shift away from the classic single-microcontroller robots to a distributed and intrinsically modular design.

Inspired by networked robots and parallel computation, Marjovi et al. presented in [4] the concept of *robotic cluster* by empowering heterogeneous robots with the

¹Hadoop is a Java based framework that supports data intensive distributed applications running on large clusters of computers.

Fig. 1 An illustrative example of the workload distribution in a multi-robot system composed of $K = 5$ robots



ability of sharing their processing resources when solving complex collective problems. A distributed multi-robot system for topological map merging was proposed. Communication between agents relied on a mesh wireless ad hoc network, and a parallel implementation based on Message Passing Interface (MPI) routines was described. Robots are equivalent to nodes of the mobile network, which locally map an environment using a topological SLAM approach, and in such a dynamic system, they continuously connect and disconnect from the mesh network, being able to discover close-by processing units in the *robotic cluster*, so as to merge local maps into global consistent topological maps.

In this work, the proposed implementation represents an evolution of the original “robotic cluster” concept by using MPI routines for parallel computers in order to implement a distributed system with lightweight loosely coupled message-queues. The use of message-queues enables us to build a less rigid system and choose from different messaging design patterns. We propose to allocate computational resources available in the multi-robot system, as well as using resources located outside the robotic systems, such as sensors of a WSN, in a similar way and without the need to use proxies to enable the communication. In order to further optimize the approach, we also explore multiprocessor architectures by applying multithreading in the computation nodes to parallelize the RBPF execution, similarly to our preliminary work presented in [20].

Hereafter, we clarify the contributions of this work to the state of the art and before describing the proposed architecture, we present the rationale behind the choice of a Gridmap RBPF SLAM approach as the key case study.

2.1 Statement of Contributions

Building upon our previous works [20, 21], the main goal of this chapter is to describe and discuss a parallel computation architecture to speed up complex steps of the

widely used GMapping SLAM approach in a distributed multi-robot system (cf. Fig. 1) or WSN, for the first time as far as our knowledge goes. Based on the *Robotic Cluster* concept, no network infrastructure is assumed to exist and robots are endowed with limited computation capabilities. This enables the system to run a distributed version of the algorithm that can cope with computation peaks. Using this approach, a system of small computers with low energy consumption is allowed to share their computation resources to solve heavy algorithm steps that previously needed help from external computers. The solution designed is independent of connections to an outside network to avoid possible communication bottlenecks and enable team scalability. Moreover, the balanced utilization of the processing resources decreases the computation load in the robot's CPU, thus reducing battery usage and improving response times.

Besides presenting a hybrid (distributed and multithreaded) architecture that allows robots to assist their peers in the SLAM task, we intend to contribute to the state-of-the-art by quantitatively analyzing the benefits of such architecture using common computers found in robots. We provide a performance and CPU load analysis to show that a significant gain in efficiency can be obtained, which leads to a more balanced utilization of the processing resources and enabling us to raise the computation load, i.e. the number of particles, up to values that would not be possible in the classical single threaded solution. Hence, this leads to significant gains in localization precision and mapping accuracy, which are quantified using a SLAM benchmarking metric. Results are extracted from frequently used SLAM datasets available in the Robotics community, in a static WSN configuration, and a real world testbed is described to show the potential of using the proposed architecture with physical teams of cooperative mobile robots, in a mobile WSN configuration.

3 Distributed Computing Architecture for RBPF SLAM

3.1 The SLAM Case-Study

SLAM has gained increasing attention over the years, and today several approaches exist to address the problem with different levels of success. Promising results to extend single robot SLAM to multiple robots have also been presented recently [22, 23]. However, solutions for the Multi-Robot SLAM (MRSLAM) problem are not equally matured to single robot SLAM since they involve several additional challenges like *rendez-vous* and robots' mutual detection, extracting relative poses between robots, conducting map alignment and merging, network latency, pose consistency in local and global maps, managing different coordinate systems, and guaranteeing updated and synchronized information in all robots.

Even though it is theoretically faster to map an environment by adopting a MRSLAM method, it is also much more computationally demanding than single robot SLAM, due to the above challenges and the fact that each robot is concurrently

building its own local map, as well as estimating its pose in the map. Furthermore, there are no widely acknowledged solutions available in the community. Therefore, we argue that using a MRSLAM approach in robots with limited computational power, e.g. swarm robots, is currently not feasible. In this work, we follow an alternative philosophy, where one of the robots in the team performs SLAM and asks assistance from its teammates to solve heavy steps of the SLAM algorithm in parallel. Advantages of the described philosophy include not having to address the additional challenges of multi-robot SLAM, and freeing the remaining robots in the team to perform other tasks in the shared environment, such as detection of people, gas leakages or other events. Also, due to the computing peaks related with heavy steps of the SLAM algorithm, single processing systems have the disadvantage of requiring a computer that is commonly oversized for its purposes.

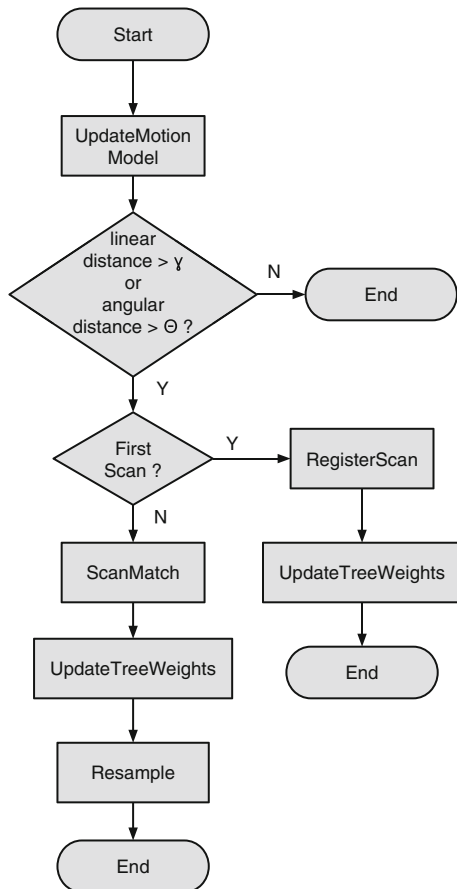
Depending on its structure, it may not be possible to parallelize the algorithm or the resultant speedup may not match the expectations, being more beneficial to run multiple instances of the algorithm with different parameters. GMapping is a grid-based RBPF implementation for the SLAM problem. It maintains an occupancy grid map divided in cells, which represent whether the state of the corresponding space is occupied (e.g. an obstacle), free (open space) or unknown. Furthermore, since it uses a PF implementation, each particle encodes a pose of the robot and the corresponding map, thus providing an hypothesis to the localization and mapping problem at a given time step.

There are several advantages for choosing GMapping as the key SLAM approach addressed in this study. Firstly, it is an open source ready-to-use algorithm available in Robot Operating System (ROS) [24]. Secondly, it is well-established with recognized performance as proven by recent works, e.g. [25]. Finally, the nature of the algorithm is particularly suitable for parallelization, given that it is a single threaded approach where each particle is computed independently, and there is no need to share data between them. This is known as an embarrassingly parallel problem. For more details on the original GMapping algorithm the reader should refer to [3].

Every time a new laser scan is acquired by the robot, the algorithm follows all the steps illustrated in the flowchart of Fig. 2. Using a profiling tool,² it was possible to verify that on average, 98.47% of the computation time of the *ProcessScan* function is spent in the scan matching step. This occurs because scan matching involves comparing the set of data returned from the Laser Range Finder (LRF) with the 2D map obtained thus far, which is a process that is repeated for all the N particles involved. Therefore, it is executed many times during localization and involves several mathematical operations, representing a high computational burden for any SLAM algorithm. Additionally, it should be noted that scan matching only occurs when the distance traveled by the robot surpasses a predetermined linear threshold γ or an angular threshold θ (cf. Fig. 2), which are important parameters of the algorithm. Hence, we expect that frequent computation peaks will occur every time the condition is verified. This is in fact shown in Fig. 3a, where a run of the algorithm in an Asus EeePC 901 netbook with $N = 10$, $\gamma = 1.0$ m and $\theta = 0.5$ rad is depicted.

²Callgrind, available in the Valgrind distribution: <http://valgrind.org>.

Fig. 2 Flowchart of the *ProcessScan* function of the GMapping algorithm, which is called every time a new scan is retrieved



While in Fig. 3a the algorithm is run with a low number of particles, in Fig. 3b we present a similar chart with $N = 40$. In this situation, one cannot distinguish the computational peaks, because scan matching takes too long. Therefore, when it is necessary to proceed with the next scan matches, the previous ones still have not finished processing, and consequently, the algorithm skips steps. This has a tremendous impact on localization and mapping, as shown later on. Consequently, in this case the algorithm will only run in real-time if the robot moves slowly, giving the computer enough time to process laser scan matches.

Thus, our preliminary study of the GMapping algorithm not only led us to parallelize the scan matching step, by distributing the N particles over the available processor resources, but also to keep the other steps as single threaded, since the overhead of parallelization together with their low computational demand would render the parallelization inefficient in these steps. In the next section we propose a distributed and multithreaded architecture for the GMapping RBPF SLAM in a solidary multi-robot system or WSN.

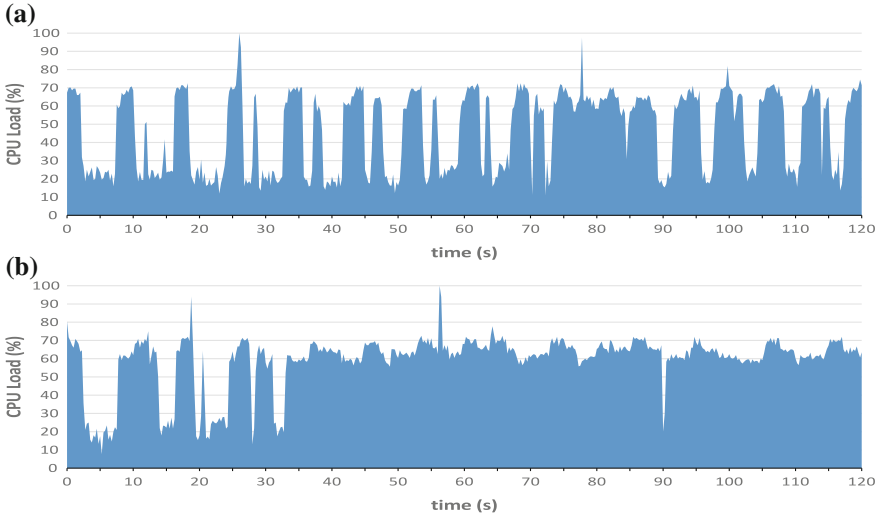


Fig. 3 CPU load in the classical GMapping algorithm running on an Asus EeePC 901 with different number of particles N . **a** GMapping with $N = 10$, **b** GMapping with $N = 40$

3.2 Hybrid Multithreaded and Distributed RBPF SLAM Architecture

A hybrid architecture was implemented based upon the GMapping algorithm so as to distribute and parallelize the laser scan matching step. In our approach, deemed as *stateful*, the wireless nodes or teammates that provide assistance run a remote process, and the main SLAM process is running on the local robot. The approach aims to minimize the data exchange between local and remote processes, by having robots sharing their context when the workload that comes from the robot running the SLAM process is distributed. The architecture is described in more detail below.

The *Stateful* architecture distributes the different particles used in the GMapping algorithm between the available external computing nodes of the robotic cluster or WSN (remote workers), and keeps a local synchronized copy at the same time—the *particle container*. This design choice allows the system to recover from faults in the network by processing data locally in such cases, and only send data to the remote workers when they are again available. In addition, keeping a local copy of all particles also simplifies the algorithm when the resampling step occurs, i.e. having all the particles, the main system can keep the trajectory tree updated and send the missing particles to the remote workers when resampling. Thus, there is no need to exchange information between different workers in the system, allowing a reduction of data flow across the network by maintaining a common state between distinct computation nodes.

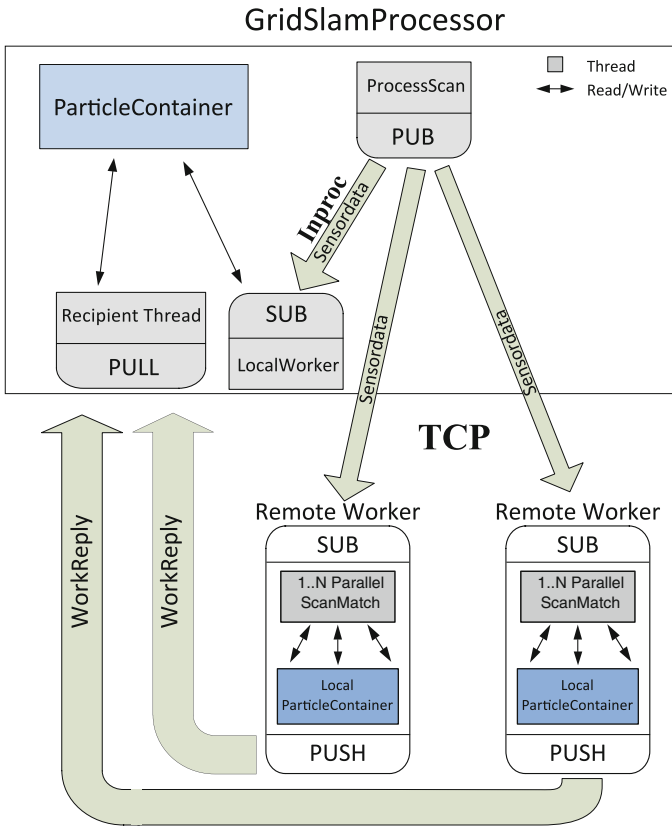


Fig. 4 Stateful MT architecture

All messages, illustrated by the green arrows in Fig. 4 are exchanged using Google's Protocol Buffer library for serialization and encapsulation,³ and the data is broadcasted to the local and remote workers, using a Publisher-Subscriber (PUB-SUB) queue. After receiving the message, each worker runs the scan matching step on the assigned particles. The local worker uses in-process communication, having direct access to the shared memory from the main system, therefore it does not need to serialize and transmit data structures. The communication with the remote workers is done via TCP.

In order to keep the particles synchronized, the remote workers send the result of the laser scan matching step to the main system as soon as it becomes available using a Push-Pull queue. This response contains a partial map, which only includes the area of the map where changes occur. A recipient thread in the main system receives the processed particles and updates the *particle container*, hence guaranteeing the consistency of the information in the whole system.

³<https://code.google.com/p/protobuf/>.

The algorithm, which follows the execution sequence illustrated in Fig. 4, maintains a common state and contextualizes the particles processed in all modules of the system. Besides the synchronization that occurs after the scan matching process, there is also a need for synchronization in the resampling process (cf. Fig. 2). In one hand, if resampling is not conducted, the main system sends an empty message to the remote worker to carry on with the algorithm execution. On the other hand, if the resampling process occurs, the main system executes this step and sends the updated particle indexes to the remote workers. In order to maintain consistency in the whole system, the remote workers may ask for missing particles that they do not possess after the resampling step. In this situation, the main system will send a compressed message with the missing particles to the remote workers using the Gzip algorithm so as to reduce the data transmitted over the network. This mechanism guarantees that all the workers in the system become synchronized.

Despite theoretically reducing the network load, maintaining a *Stateful* architecture comes at a cost: there are no means for automatic load balancing. The workload, i.e. the number of particles, must be distributed in the beginning.

In order to exchange information between computers the following messages are used in the *Stateful* approach:

- **StartPackage:** Initial message sent to the remote workers, contains all the parameters needed for the laser scan matcher and the motion model used. This is sent only in the beginning of the algorithm.
- **Sensordata:** Message with the sensor data sent at each iteration of the algorithm. Contains updated pose data for every particle and data from the Laser Range Finder (LRF).
- **WorkReply:** Message with the index of the processed particle, the new pose and the new weights, the area changed and optionally the new size of the map, in case it grew.
- **IndexMessage:** Message sent by the main system after the resampling process. Contains the indexes of all valid particles and is sent to every remote worker.
- **ResampleMessage:** Message used to respond to the IndexMessage. It contains the indexes of the local particles that are missing by the remote worker.
- **Particles:** Message sent by the main system after a ResampleMessage. It contains the particle's data (map, pose and weights).

In addition to distributing the particles over the local and remote workers, the proposed architecture also leverages from multithreading to use the computational resources in an even more efficient way. Thus, our method divides the N particles in equally sized chunks and distributes them over the available processor cores of each worker.

In order to implement the *Stateful* multithreaded (MT) parallel architecture, we have used the OpenMP programming module. OpenMP is supported by a large number of compilers on different platforms, providing a simple and elegant solution to the single threaded limitation of the original algorithm, without the need to restructure and modify the existing code or use external libraries. This approach has the ability to autodetect the number of available processor cores, also providing an option

to choose the number of worker threads. This way, the algorithm adapts itself to the host architecture, making an efficient use of the processing resources. Furthermore, this approach does not limit its use to GMapping, having also the potential to be used in any SLAM algorithm that relies on laser scan matching in a similar particle-independent way.

The benefits of using a hybrid architecture that employs both particle distribution over the available computing nodes and multithreading inside each node is discussed in the next section. Note also that the implementation of the architecture was done using the Zero Message Queue (ZeroMQ) library,⁴ which enables the use of different types of queues such as publisher-subscriber, request-reply or push-pull, as well as supporting several types of transport, e.g. in process, inter-process, TCP, multicast, etc. The open-source implementation of the hybrid architecture for the GMapping SLAM approach is publicly available for download.⁵

4 Results and Discussion

In this section, experimental results are presented and discussed. We propose to analyze the efficiency gain, and the localization and mapping performance using the proposed architecture. Having this in mind, we have tested 3 datasets typically used in SLAM benchmarking: ACES Building, Intel Research Lab, and MIT CSAIL Building,⁶ in a static WSN configuration. Further experiments were conducted with physical teams of mobile robots to validate the system in a real world scenario, in a mobile WSN configuration.

The experimental tests that were run on the SLAM benchmarking datasets make use of the two distinct computer architectures depicted in Table 1. The EeePC runs the main SLAM system (local worker) and Odroid X2 single-board computers are used as remote workers. These were chosen in order to validate the approaches proposed using heterogeneous processors with distinct computation power, having at the same time a main system that is very limited in terms of computation power. In these experiments, the computers were connected via Ethernet to a 100 Mbps switch.

In these tests, several different configurations were tested. The number of particles of the GMapping algorithm were increased from $N = 30$ until $N = 120$ in steps of 30, eventually stopping when an “out of memory” error occurs in the Asus EeePC 901. The total number of local and remote workers in the system K is increased up to a maximum of 3. Furthermore, for each different configuration we run 5 trials, resulting in a total of 275 trials, with the following combinations:



1. $dataset = \{ACES\ Building, Intel\ Research\ Lab, MIT\ CSAIL\ Building\}$,
2. $N = \{30, 60, 90, 120\}$,

⁴<http://zeromq.org>.

⁵<https://github.com/brNX/gmapping-stateful>, Hybrid branch.

⁶<http://kaspar.informatik.uni-freiburg.de/~slamEvaluation/datasets.php>.

Table 1 Specifications of the Asus EeePC netbook and the Odroid X2 single-board computer used in the experiments

Computers Used	 Asus EeePC 901	 Odroid X2
CPU	Intel Atom™ N270 with Hyper-Threading	Exynos4412 Prime ARM Cortex-A9 Quad Core
# Cores	1 Physical (2 Virtual) Cores	4 Physical Cores
Clock Speed	1.6 GHz	1.7 GHz
Cache L2	512 KB	1 MB
RAM	1GB (DDR2)	2GB (DDR2)
Storage	20GB SSD	16GB, SD card
Operating System	Ubuntu 12.04, 32 bit	Ubuntu-based Linaro 12.11, 32 bit
Approximate Cost	\$219	\$135

3. $architecture = \{Classical, Stateful\ ST, Stateful\ MT\}$,

4. $K = \{1, 2, 3\}$,

where ST refers to the single threading version in the computing nodes, and MT to the multithreaded version.

4.1 Efficiency Gain

We start by analyzing the average time taken for the laser scan processing step of the SLAM algorithm. To this end, we have used the classical single processing GMapping approach, and the hybrid *Stateful* MT architecture proposed herein. Table 2 presents the average time in seconds over the 5 trials with every distinct configuration. Note that there are no results for $N = 120$ in the ACES Building, due to the 1 GB memory limitation in the Asus EeePC 901.

In terms of load balancing in the proposed architecture, 10 particles were always assigned to the main system, while the remaining particles were equitably distributed over the $K - 1$ remote workers. This value was conservatively chosen to guarantee that no scan matching steps are skipped in the main system running on the Asus EeePC, and we leave the optimization of particles assignment as future work.

The first immediate evidence is that scan processing takes much less time in the *Stateful* architecture than in the classical one in all tested cases. Additionally, it becomes clear that multithreading plays an important role in the scan matching

Table 2 Average laser scan processing times (in seconds) over 5 trials for each different combination $\{\text{dataset}, N, \text{architecture}, K\}$

	N \ K	Classical	<i>Stateful</i>			
		1	2(ST)	2(MT)	3(ST)	3(MT)
ACES Building, Austin $\approx 56 \times 58$ (m)	30	1.417	0.616	0.663	0.806	0.781
	60	2.797	1.053	0.645	0.864	0.822
	90	4.226	1.825	0.661	1.507	1.161
Intel Research Lab, Seattle $\approx 28.5 \times 28.5$ (m)	30	1.557	0.607	0.644	0.713	0.684
	60	3.102	1.109	0.681	0.702	0.680
	90	4.653	1.882	0.649	1.068	0.774
	120	6.129	2.604	0.807	1.469	0.722
MIT CSAIL Building, Cambridge (MA) $\approx 61 \times 46.5$ (m)	30	3.020	1.218	1.202	1.492	1.424
	60	6.007	2.141	1.258	1.482	1.392
	90	8.939	3.624	1.315	2.227	1.607
	120	11.942	4.973	1.546	2.912	1.497

process, as the *Stateful* MT version generally leads to less processing time when compared to the single threaded one. The proposed approach accelerates the processing time by up to a factor of 6.39 (ACES), 8.49 (Intel) and 7.98 (CSAIL), and some relevant global trends can also be observed. For instance, in the CSAIL dataset the processing time is larger than in the other datasets. This happens due to the resolution of the laser scan messages. In the CSAIL dataset there are 360 beams in each scan, while in the other datasets there are only 180. In addition, the scan processing time generally drops with the number of remote workers, due to the progressive decrease of computational load on each worker when more processors are involved. However, this is not necessarily true, especially when there are too many workers K for few particles N , i.e. when the ratio N/K is low. In such situations, since each worker only has a small portion of the total number of particles, when resampling occurs the probability of the remote workers to ask for missing particles increases. Waiting for the missing particles to be sent over the network has an impact on the time to process the scans, and clearly delays in the system increase with the number of particles requested. Hence, deceleration even occurs if the *Stateful* architecture is not run within specific intervals of N/K .

As a consequence of the above facts, it is possible to run a distributed and multi-threaded method with several more particles than the classical method, thus gaining in localization and mapping quality. A clear example of this is shown in the results for the ACES building, where the classical method with 30 particles takes approximately the same amount of time to process the laser scans as the *Stateful* ST architecture with 60 particles and 2 workers, or the *Stateful* MT architecture with 90 particles

and 3 workers. Furthermore, since computation sharing accelerates the scan matching step it enables the robot that is collecting data to move faster. When moving at higher speeds, the algorithm reaches the γ or the θ threshold quicker. Performing SLAM with GMapping in real-time without dropping data is possible as long as the processing time is shorter than the time between the thresholding condition is verified.

In the analysis presented before, we have only addressed the time to process laser scans. However, it is important to understand how the overall time to run the algorithm has been accelerated. In parallel computing, it is common to use the speedup metric to measure how fast a parallel algorithm is, when compared to the corresponding sequential algorithm. Therefore, speedup ν is defined as:

$$\nu = \frac{T_{Class}}{T_{Hyb}}, \tag{1}$$

where T_{Class} represents the total processing time of the classical method, and T_{Hyb} the total processing time of the hybrid method in the same exact conditions. In Fig. 5, the evolution of the speedup metric using the hybrid *Stateful* MT architecture with increasing number of particles is shown. The illustrated curves confirm that the efficiency gain tends to increase in general with processing load, i.e. the number of particles N . This is common in distributed computation because the overhead of parallelization, such as the creation and management of threads, is approximately the same independently of the size of the problem. Therefore, the ratio between overhead and workload decreases when the processing load grows. As a result, the efficiency tends to increase with the size of the problem. This is more evident when several missing particles are not required in the resampling step. However, when this is the case, as for example shown in the ACES building with up to 90 particles, the speedup when using $K = 2$ workers is larger than with $K = 3$.

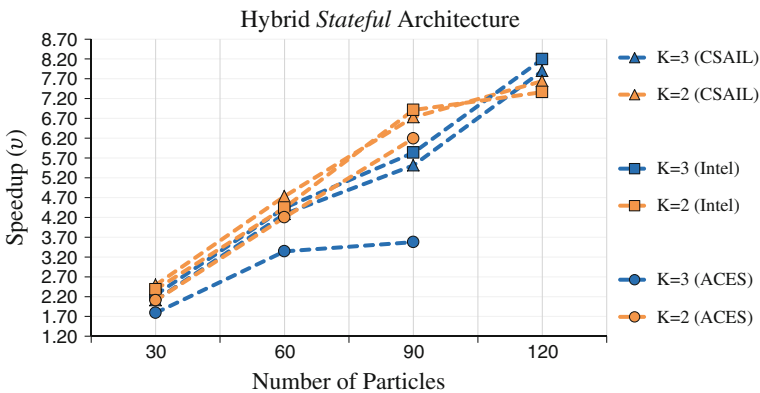


Fig. 5 Speedup using the proposed architecture

4.2 SLAM Performance Analysis

Due to the acceleration observed in the scan processing step, it has been shown that the proposed architecture is able to handle a number of particles that cannot necessarily be handled in the classical method without dropping data, as shown previously in Fig. 3. Hereupon, the performance obtained with the hybrid architecture is expected to be superior in situations where the classical approach skips scan matching steps, having at least the same performance levels when this is not the case. This is clearly noticed in the Intel Research Lab dataset with $N = 90$ particles (*cf.* Fig. 6).

Despite the evident increase of mapping quality, visual inspection of the resulting maps does not allow a detailed comparison. So, the need to precisely evaluate the results asks for a more accurate method—a quantitative scale. In this work, we make use of the benchmarking metric presented in [26] to assess the impact of computation sharing in SLAM performance. This metric evaluates the accuracy of the poses in robot trajectories during data acquisition. It uses only relative *relations* between poses and does not rely on a global reference frame, which even allows to compare algorithms with different estimation techniques and sensor modalities.

The mean translational error, $\bar{\epsilon}_{trans}$, and the mean angular error, $\bar{\epsilon}_{rot}$, with the corresponding standard deviation were calculated for the maps generated by the classical and *Stateful* MT approaches (*cf.* Fig. 6a, c). Additionally, in the chart of Fig. 7, one can see the evolution of the translational error over the different pose *relations* in a logarithmic scale. Results clearly demonstrate that the errors of the proposed architecture are inferior to those of the classical method, both for the translational and rotational error. The peaks in the evolution chart show situations where relations measured by the classical method have high errors, which explains why the map generated was only consistent locally, but tends to be inconsistent as a whole.



Fig. 6 Mean error in the Intel Research Lab dataset with $N = 90$. **a** Classical approach. $\bar{\epsilon}_{trans} = 0.065 \pm 0.38$ (m), $\bar{\epsilon}_{rot} = 2.809 \pm 6.539$ (deg), **b** *Stateful* MT approach with $K = 2$. $\bar{\epsilon}_{trans} = 0.028 \pm 0.037$ (m), $\bar{\epsilon}_{rot} = 1.977 \pm 1.817$ (deg)

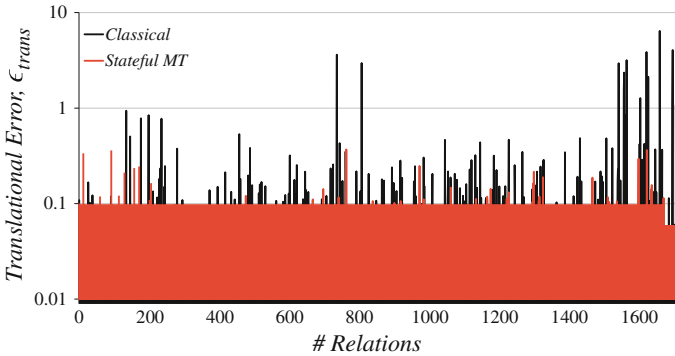


Fig. 7 Translational Error, ϵ_{trans} (logarithmic scale)

Results have shown that the hybrid architecture proposed leads to extremely efficiency gain in a SLAM task especially when properly dimensioned both in terms of N/k and computation load. Also, it enables significant performance gains. Furthermore, since it limits the flow of data in the network, the proposed approach requires a minimal amount of memory in the main system. On the downside, occasional delays may occur due to the transmission of particles to the local workers.

4.3 Real World Experiment with a Team of Robots

Interesting advantages of using a hybrid SLAM architecture has been demonstrated using a static WSN configuration, where the robot that is performing the SLAM task may be assisted by external computing nodes in the scan matching step of the algorithm. However, it is crucial to validate the work in real world *Robotic Clusters*, where the computation resources are mobile, the wireless communication connectivity is unpredictable and yet the system must perform in real-time in a mobile WSN configuration. Having this in mind, a large arena was built in a class room of the Department of Electrical and Computer Engineering of the University of Coimbra, which is shown in Fig. 8a, and a ground truth map of the arena was designed as a reference (cf. Fig. 8b). The robots used to perform the experiment were a team of three ROS-enabled iRobot Roombas shown in Fig. 8c. These are equipped with an Asus EeePC 901 with very limited processing capabilities, and an Hokuyo URG-04LX LRF with a 10Hz scan rate.

The experimental test consisted in having the three robots exploring the scenario in leader-follower motion. The robot in the front, i.e. the leader, acquired sensor data and ran a distributed version of the GMapping algorithm using the *Stateful MT* architecture with $N = 30$ particles, while being assisted by the other two robots, i.e. the followers, which made their processing resources available, thus serving as remote workers. The particles were equally distributed by each of the $K = 3$ workers.

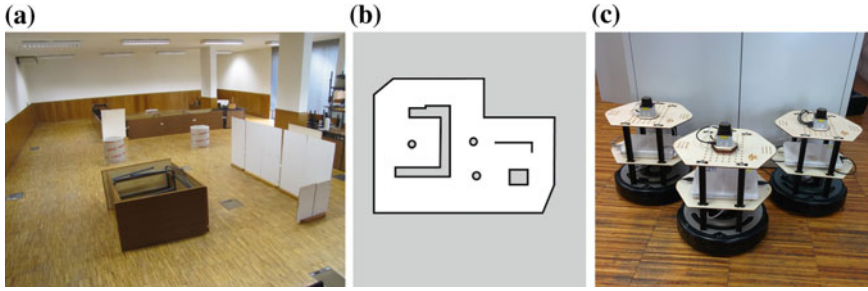


Fig. 8 Experiments using a Robotic Cluster in a mobile WSN configuration. **a** Experimental scenario 12.85×9.15 m, **b** ground truth map, **c** robots used

The robots made use of their onboard 802.11b/g/n wireless card for communication, and moved at a maximum linear speed of 0.4 m/s, and angular speed of 0.5 rad/s.

The map generated in real-time using the *Stateful MT* architecture is illustrated in Fig. 9a. As can be seen when compared to the ground truth, the map is generally consistent and resembles the ground truth reference with some glitches. In order to be able to compare the generated map with the classical approach, all sensor data were recorded during the experiment. This allowed to run the classical GMapping approach offline with $N = 30$ in the Asus EeePC 901, being fed with the exact same laser data. The resulting map is presented in Fig. 9b. Once again, the advantage of parallelization and distributing the computation load of the SLAM algorithm becomes clear by comparing the consistency of both maps

In these tests, the average laser scan processing time of the classical approach was 1.317 and 0.461 s in the hybrid approach ($\simeq 2.9$ times faster). However, it was not possible to conduct a numeric performance analysis, since there is no ground truth of *relations* between poses of the robot during the experiment.

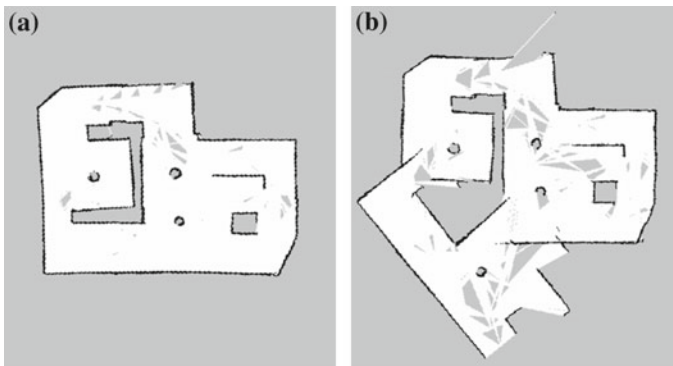


Fig. 9 Maps generated in the real world experiment. **a** *Stateful MT* architecture $N = 30$, $K = 3$, **b** classical approach $N = 30$

5 Conclusion

This work presents a hybrid distributed and multithreading architecture based on lightweight loosely coupled message-queues to speed up a 2D RBPF SLAM approach, widely used by the Robotics community. The proposed architecture enables a robot to be assisted by several static computing nodes, as well as a solitary multi-robot system to share its limited computation resources in order to solve a complex SLAM problem without depending on connections to an outside network. This leads to a balanced utilization of resources, reducing battery usage and response time in computing nodes, and also fosters team scalability, by avoiding communication bottlenecks to a central entity.

Several experimental tests were conducted using diverse configurations in common benchmark datasets, which demonstrated that processing time drastically decreases when the computational resources are used efficiently. In addition, it was shown the increase of localization and mapping performance when the number of particles cannot be fully handled by the single processing solution. Advantages and disadvantages of the hybrid architecture have been discussed, and a real world testbed was described to pinpoint the potential of using the proposed philosophy in teams of physical robots.

Higher values of speedup ($v > 8$) than those obtained in this work could eventually be reached using more powerful processors. However, this stands outside the scope of the work, as we proposed to utilize economic, small-sized and limited processing components frequently used in robotic platforms. In the future, it would be interesting to adapt the load distribution online in the hybrid approach, e.g. by taking into consideration the history of processed particles by each worker in the system; and also assess the intervals of N/k where optimized performance of the approach can be attained. Furthermore, despite not being tested in this paper, the authors suggest that similar architectures can be used in the future to speed up any SLAM algorithm that relies in laser scan matching.

References

1. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: AAAI National Conference on Artificial Intelligence (2002)
2. Agarwal, P., Tipaldi, G.D., Spinello, L., Stachniss, C., Burgard, W.: Robust map optimization using dynamic covariance scaling. In: Proceedings of the International Conference on Robotics and Automation (ICRA 2013), Karlsruhe, Germany, May 6–10 (2013)
3. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Trans. Robot.* **23**(1), 34–46 (2007)
4. Marjovi, A., Choobdar, S., Marques, L.: Robotic clusters: multirobot systems as computer clusters: a topological map merging demonstration. *Robot. Auton. Syst.* **60**(9), 1191–1204 (2012)
5. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. *IEEE Robot. Autom. Mag.* **13**(2), 99–110 (2006)

6. Dissanayake, D., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robot. Autom.* **17**(3), 229–241 (2001)
7. Thrun, S., Montemerlo, M.: The graphSLAM algorithm with applications to large-scale mapping of urban structures. *Int. J. Robot. Res.* **25**, 403–429 (2006)
8. Kohlbrecher, S., Meyer, J., Von Stryk, O., Klingauf, U.: A flexible and scalable SLAM system with full 3D motion estimation. In: *Proceedings of the International Symposium on Safety, Security and Rescue Robotics* (2011)
9. Pedrosa, E., Lau, N., Pereira, A.: Online SLAM based on a fast scan-matching algorithm. In: Correia, L., Reis, L.P., Cascalho, J. (eds.) *Progress in Artificial Intelligence, Lecture Notes in Computer Science*, vol. 8154, pp. 295–306, Springer, Berlin (2013)
10. Machado Santos, J., Couceiro, M.S., Portugal, D., Rocha, R.P.: A sensor fusion layer to cope with reduced visibility in SLAM. In: *Journal of Intelligent and Robotic Systems (JIINT), Special Issue on Autonomous Robot Systems*, Springer, London (2015)
11. Chandrakasan, A.P., Potkonjak, M., Mehra, R., Rabaey, J., Brodersen, R.W.: Optimizing power using transformations. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst.* **14**(1), 12–31 (1995)
12. El Hamzaoui, O., Steux, B.: A fast scan matching for grid-based laser SLAM using streaming SIMD extensions. In: *Proceedings of the 11th International Conference on Control, Automation, Robotics and Vision (ICARCV 2010)*, Singapore, 7–10 Dec 2010
13. Clipp, B., Lim, J., Frahm, J., Pollefeys, M.: Parallel, real-time visual SLAM. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3961–3968, Taipei, Taiwan, Oct 2010
14. Par, K., Tosun, O.: Parallelization of particle filter based localization and map matching algorithms on multicore/manycore architectures. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. Baden, Germany, June 2011
15. Miah, M.S., Bolic, M.: Parallel implementation of modified rao-blackwellised particle filter. University of Ottawa, Technical Report (2009)
16. 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)
17. Arumugam, R., Enti, V.R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F.F., Kumar, A.S., Meng, K.D., Kit, G.W.: DAVinCi: a cloud computing framework for service robots. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3084–3089, Anchorage, Alaska, USA (2010)
18. Siegwart, R., Balmer, P., Portal, C., Wannaz, C., Blank, R., Caprari, G.: RobOnWeb: a setup with mobile mini-robots on the web. In: *An Introduction to Online Robots*, MIT Press, In Beyond Webcams (2002)
19. Dorigo, M., Floreano, D., Gambardella, L.M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., et al.: Swarmanoid: a novel concept for the study of heterogenous robotic swarms. *IEEE Robot. Autom. Mag.* **20**(4), 60–71 (2013)
20. Gouveia, B.D., Portugal, D., Marques, L.: Speeding up rao-blackwellized particle filter SLAM with a multithreaded architecture. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, USA, 14–18 Sep 2014
21. Gouveia, B.D., Portugal, D., Marques, L.: Computation sharing in distributed robotic systems: a case study on SLAM. *Proc. IEEE Trans. Autom. Sci. Eng., Spec. Issue Cloud Robot. Autom.* **12**(2), 398–409 (2015)
22. Lazaro, M., Paz, L., Piniés, P., Castellanos, J., Grisetti, G.: Multi-robot SLAM using condensed measurements. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1069–1076. Tokyo, Japan, 3–7 Nov 2013
23. Eich, M., Hartanto, R., Kasperski, S., Natarajan, S., Wollenberg, J.: Towards coordinated multi-robot missions for lunar sample collection in an unknown environment. *J. Field Robot.* **31**(1), 35–74 (2014)
24. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, R.: ROS: an open-source Robot Operating System. In: *International Conference on Robotics and Automation*, WS on Open Source Software, Kobe, Japan (2009)

25. Machado Santos, J., Portugal, D., Rocha, R.P.: An evaluation of 2D SLAM techniques available in robot operating system. In: Proceedings of the: International Symposium on Safety, Security and Rescue Robotics, Linköping, Sweden (2013)
26. Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., Kleiner, A.: On measuring the accuracy of SLAM algorithms. *Auton. Robots* **27**(4) (2009)

A Particle Filter-Based Method for Ground-Based WSN Localization Using an Aerial Robot

Francisco Cuesta, Miguel Cordero, Luis Díaz, Antidio Viguria
and Aníbal Ollero

Abstract A particle filter-based algorithm for 3D localization of WSN (Wireless Sensor Network) nodes from an Unmanned Aerial Vehicle (UAV) is proposed. The algorithm uses the information obtained from RSSI measurements taken by a node located on-board the UAV together with its estimated position provided by the UAV navigation system. As the WSN nodes are assumed to be on the ground, Digital Elevation Model (DEM) data has been used to improve the accuracy of the estimation. WSN localization algorithms have typically been validated in short range scenarios. In this paper, field experiments have been carried out to validate the proposed algorithm in medium range scenarios. For this purpose real flights have been conducted with a fixed-wing UAV flying up to 1.5 km away from the transmitter to be located. During the flights the algorithm was running on an on-board embedded computer and the estimated position was sent to the ground control station for monitoring purposes. The results of these experiments are presented in this paper.

1 Introduction

Most of the localization algorithms proposed in literature for WSN (Wireless Sensor Network) makes use of the Received Signal Strength Indication (RSSI) [1]. The main reason is that pure RSSI methods can be readily deployed in every WSN, since

F. Cuesta · M. Cordero (✉) · L. Díaz · A. Viguria
Center for Advanced Aerospace Technologies, 41309 La Rinconada (Seville), Spain
e-mail: fcuesta@catec.aero

M. Cordero
e-mail: mcordero@catec.aero

L. Díaz
e-mail: ldiaz@catec.aero

A. Viguria
e-mail: aviguria@catec.aero

A. Ollero
University of Seville, 41092 Seville, Spain
e-mail: aollero@us.es

the RSSI circuitry is natively supported by most of the existing transceiver chipsets, without the need of extra hardware. This reduces the cost and energy consumption with respect of other localization methods (e.g. using multiple antennas). In general, WSN node localization algorithms assume the presence of a limited number of reference nodes in the network. These nodes know their own position and are called beacon or anchor nodes [1]. This information is used to estimate the location of the rest of nodes in the network. Broadly speaking, RSSI-based localization algorithms can be divided in two wide categories [2]:

- Range-free (or proximity-based) approaches, which infer constraints on the proximity to beacon nodes. These methods are quite simple but they have inherently limited precision.
- Range-based approaches, which rely on range measurements to compute the position of the nodes and they can provide better accuracies than range-free approaches.

RSSI-based localization algorithms have been widely used for Wireless Sensor Network localization in indoor (e.g. comparative surveys in [1, 3]) and outdoor environments (see e.g. [4, 5]). In outdoor environments, GNSS (Global Navigation Satellite Systems) systems are extensively used in many applications. However, due to restrictions on cost, size and power consumption, GNSS receivers cannot generally be used to locate all the nodes in a WSN [6] and alternative approaches such as RSSI-based localization methods are preferred. In these cases, GNSS receivers can be used to estimate the position of a limited number of beacon nodes.

Most of the RSSI-based localization algorithms proposed for outdoor applications are based on deterministic methods that assume that range estimations obtained from RSSI measurements are accurate (e.g. trilateration methods [4, 5]). Although these types of algorithms are generally much simpler, they cannot deal with the probabilistic (noisy) nature of RSSI-based range and hence they cannot achieve optimal and robust solutions. In the last decade new RSSI-based localization algorithms using probabilistic frameworks have been presented. Many of these algorithms are based on particle filters, an approximation to recursive Bayesian estimation (see e.g. [7]).

It has already mentioned that most algorithms assume the presence of a limited number of beacon nodes. When these nodes are static, it might be necessary to deploy several of them to cover the whole WSN area. This can be avoided using mobile beacon nodes that can also act as data mules to collect sensor data from the rest of the nodes (see e.g. [8, 9]). These mobile beacon nodes can be mounted on-board a ground or aerial vehicle. The cases in which the vehicles are autonomous are of particular interest. In many applications aerial robots are preferred over ground robots because they can operate over rough terrains and they have a larger range of operation.

Particle filter RSSI-based localization algorithms have already been used to locate WSN nodes from Unmanned Aerial Vehicles (UAV). This approach has been followed in [10] where RSSI measures collected by a flying beacon-node on-board an autonomous helicopter were used for the localization of the WSN nodes. In this case the algorithm was run on a laptop PC on the ground and not on-board the UAV. In addition, the maximum distance from the UAV to the nodes to be located was about

80 m. The same algorithm was also tested with a ground autonomous vehicle [11]. In this case the algorithm was run on-board the ground vehicle. In [12, 13] a particle filter-based RSSI localization algorithm is proposed to locate animal radio tags for wildlife applications using a UAV. In this case the algorithm takes into account the radiation pattern of the antenna in the estimation process which allows improving the obtained results. However, these algorithms have only been tested on the ground and the algorithm was not run in real-time but the data was recorded and post-processed in the lab.

1.1 Contributions of the Book Chapter

In the work presented in this chapter the particle filter based localization algorithm using RSSI measurements taken from a UAV proposed in [10] has been tested in real flight experiments in which the algorithm has been run on real-time on-board the UAV. The purpose of this work is to test and validate the algorithm in extended ranges (1.5 km compared with the range of 80 m in [10]). To achieve these ranges, the sensor nodes have been substituted by a ground based WiMAX base station (acting as the node to be located) and a WiMAX subscriber station on-board the UAV. In addition, unlike in previous works (including [10, 12, 13]) in which the experimental results were based on post-processing of logged data or simulations, in this work the localization algorithm was run in an embedded computer onboard the UAV. The position of the ground-based transmitter was estimated in real time and sent to a ground station for visualization during the flight. The estimation accuracy has also been improved by incorporating terrain information from DEM (Digital Elevation Models) data which are used as a restriction for the height coordinate of the ground-based node. Unlike [12, 13], the algorithm presented in this chapter do not take into account radiation pattern of the antenna in the estimation process at this moment. The authors expect to include this improvement in the near future.

This chapter is organized as follows: in Sect. 1 the principles behind the algorithm are presented; in Sect. 2 an in-depth description of the algorithm implementation is provided; next, in Sect. 3 the experimental results are shown and analyzed; and finally some conclusions and future work are presented in the last section.

2 Statistical Principles of the Algorithm

The objective of the localization algorithm is to solve a tracking problem where the location of the ground-based transmitting node has to be estimated at time k given the UAV positions and RSSI measures up to time k . This dynamic system can be stated using a state space form with system and measurement models which are given in (1) and (2),

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \quad (1)$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \quad (2)$$

where \mathbf{x}_k is the state vector at time k , \mathbf{z}_k is the measurement vector at time k , \mathbf{v}_{k-1} is the system noise at time $k - 1$ and \mathbf{n}_k is the measurement noise at time k . The functions and $\mathbf{f}_k(\cdot)$ and $\mathbf{h}_k(\cdot)$ are assumed to be known as part of the system model.

2.1 Recursive Bayesian Estimation

A state space model can also be viewed as a Hidden Markov Model (HMM) characterized by the conditional probability density functions (PDF) $p(\mathbf{x}_k/\mathbf{x}_{k-1})$ and $p(\mathbf{z}_k/\mathbf{x}_k)$ (which is known as the likelihood function) [14]. Note these PDFs are assumed to be known as part of the system model. From a Bayesian point of view all relevant information about the state at time k given observations up to and including time k can be obtained from the posterior probability density function (PDF) of the state based on all available information (i.e. $p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_k)$). Hence, Bayesian estimation filter aims at building the posterior PDF which is considered to be the complete solution to the estimation problem.

When the estimation has to be updated every time a new measurement is available, recursive filters are a convenient solution. These filters estimate the posterior $p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_k)$ at time k from the posterior at time $k - 1$ (i.e. $p(\mathbf{x}_{k-1}/\mathbf{x}_{k-2}, \mathbf{z}_{k-1})$). This is done in two steps:

- The prediction stage in which the prior PDF (i.e. the PDF obtained before incorporating the information from the new measurement) is obtained from the previous posterior PDF using the Chapman-Kolmogorov equation given by (3).

$$p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_{k-1}) = \int p(\mathbf{x}_k/\mathbf{x}_{k-1})p(\mathbf{x}_{k-2}, \mathbf{z}_{k-1})d\mathbf{x}_{k-1} \quad (3)$$

- The update stage in which the information from the new measurement is used to modify the prior PDF to obtain the posterior PDF. This is done using the Bayes theorem given by (4).

$$p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_k) = \frac{p(\mathbf{x}_k/\mathbf{x}_k)p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_{k-1})}{\int p(\mathbf{x}_k/\mathbf{x}_k)p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_{k-1})d\mathbf{x}_k} \quad (4)$$

Note that in (3) and (4) the conditional PDFs $p(\mathbf{x}_k/\mathbf{x}_{k-1})$ and $p(\mathbf{z}_k/\mathbf{x}_k)$ that are used to get the posterior at time k from the posterior at time $k-1$ are assumed to be known as part of the Hidden Markov Model. Once the posterior PDF is estimated the estimation of the state is the mean value obtained as

$$\hat{\mathbf{x}}_k = \mathbf{u}_k = \int \mathbf{x}_k p(\mathbf{x}_k/(\mathbf{x}_{k-1}, \mathbf{z}_k))d\mathbf{x}_k \quad (5)$$

In general, recursive Bayesian filters cannot be described using analytical forms but in the particular case of linear Gaussian state space models with a known initial

state (the analytical form in this case is the Kalman Filter) [15]. However, when no knowledge about the initial state is available or the dynamic system cannot be modeled with Gaussian processes approximated methods have to be adopted.

2.2 Particle Filters

Sequential Monte Carlo (SMC) methods (also known as Particle Filters) are the most used method for approximated recursive Bayesian estimation. These methods are based on the Sequential Importance Sampling (SIS) algorithm which represents the posterior PDF by a set of random samples with associated weights as

$$p(\mathbf{x}_k/\mathbf{x}_{k-1}, \mathbf{z}_k) \approx \sum_{i=1}^{N_s} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \quad (6)$$

where $\{\mathbf{x}_k^{(i)}, i = 1, \dots, N_s\}$ are the N_s samples (which are called particles) of the state space and $w_k^{(i)}$ are the normalized weights associated with those particles (i.e. $\sum_i w_k^i = 1$).

As a recursive algorithm, each iteration the particles are updated using the approximated posterior as shown in (7).

$$\mathbf{x}_k^i \sim p(\mathbf{x}/\mathbf{x}_{k-1}^i, \mathbf{z}_k) \quad (7)$$

In addition, the weights of the particles are also updated using the likelihood function as shown in (8) [15].

$$w_k^i = p(\mathbf{z}_k/\mathbf{x}_k^i) w_{k-1}^i \quad (8)$$

The new weights have to be normalized as $w_k^{(i)} = w_k^{(i)} / \sum_{i=1}^N w_k^{(i)}$ so the cumulative probability of the posterior remains equal to the unity.

2.3 Resampling

The Sequential Importance Sampling (SIS) algorithm experiences a problem known as degeneracy phenomenon that consists on the gradual increase of the values of the bigger weights at the expense of the smaller weights. This will cause that after a few iterations all but one particle will have negligible weight so the tracking capability of the algorithm will be severely decreased. To measure the degeneracy of the algorithm, the effective sample size [16] can be estimated as

$$\hat{N}_{eff} = 1 / \sum_{i=1}^N (w_k^{(i)})^2 \quad (9)$$

To reduce the effects of degeneracy, the resampling method can be applied if the estimated effective sample size given by (4) is below a certain threshold. The resampling method proposed in [17] has been used.

Next section will describe the particle filter implementation that has been used in the transmitter localization problem.

3 Localization Algorithm

The objective of the localization algorithm is to estimate the position of a ground-based transmitter from the path loss experienced by the signal and the position of the UAV provided by its navigation system. UAV navigation use Kalman Filtering techniques to get enhanced attitude (i.e. orientation), position and velocity estimations by combining measurements from inertial sensors (gyroscopes and accelerometers), magnetic compass sensors and GPS-based position and velocity estimations [18]. When using non-differential GPS receiver these systems can provide the position of the UAV with an accuracy of about 2 m; when using differential RTK GPS systems the navigation system can estimate the position of the UAV with an accuracy of a centimeter.

The signal path loss is calculated by the on-board PC from the RSSI measurements and the values of the transmitted power which is assumed to be known (this assumption will be justified in section 0). Hence, the state vector \mathbf{x}_k to be estimated is the position of the transmitter and the measurement vector \mathbf{z}_k consists of pairs $\{L_k, \mathbf{x}_k^{UAV}\}$. The algorithm used in this paper is that proposed in [10, 11]. In the next section this algorithm will be described in detail.

3.1 Likelihood Function

The likelihood function $p(\mathbf{z}_k/\mathbf{x}_k^i)$ relates the measurement and the state vectors so in this case it relates the transmitter position with the UAV position and the signal loss measurements. As shown in previous section this function is used for updating the weights of the particles according to (8). The distance between the transmitter and the receiver has been used to relate the signal loss and the UAV position with the transmitter position. This distance can be calculated at instant k as $d_k = \|\mathbf{x}_k - \mathbf{x}_k^{UAV}\|$. The likelihood function has been assumed to be Gaussian so it can be written as

$$L_k = \mu(d_k) + N(0, \sigma(d_k)) \quad (10)$$

$$p(\mathbf{z}_k/\mathbf{x}_k^i) = \frac{1}{\sigma(d_k)\sqrt{2\pi}} \exp\left(-\frac{(L_k - \mu(d_k))^2}{2\sigma^2(d_k)}\right) \quad (11)$$

It is well known that a correlation exists between the signal loss at the receiver and the distance between it and the transmitter. This is clear from the Friis transmission

equation which calculates the signal strength at the output of a receiver assuming free space propagation [19]. This equation is given in logarithmic form by

$$P^{(r)} = P^{(t)} + G_t + G_r + 20 \log_{10}\left(\frac{\lambda}{4\pi d}\right) \quad (12)$$

where $P^{(r)}$ is the received power, $P^{(t)}$ is the transmitted power, λ is the wavelength, d is the distance between the transmitter and receiver and G_r and G_t are the antenna gain of the receiver and transmitter in the direction between them. If the signal loss is defined as the difference between the transmitted and the received power, i.e. $L = P^{(t)} - P^{(r)}$, (12) can be rewritten as

$$L = 20 \log_{10}(4\pi d) - 20 \log_{10}(\lambda) - G_t - G_r \quad (13)$$

$$L = 20 \log_{10}(4\pi d) - 20 \log_{10}(\lambda) \quad (14)$$

In the current version of the algorithm the antenna gains have not been taken into account in the calculation of the signal loss so Eq. (14) has been used for calculating the signal loss instead of (13). Hence, the effects of the antenna gains have not been removed from the signal loss calculation causing an extra error in the position estimation. It is important to note that antenna patterns have different gains depending on the direction between the transmitter and the receiver as well as their orientations with respect to each other. Hence, the values of the antenna gains that appear in (13) are not constant but changes according to the relative movement between the UAV and the ground transmitter. In order to minimize the influence of the antenna gains in the results, the flight paths have been designed with long straight segments in order to reduce the turn angles and the effects of the antenna gains.

As can be seen from (13), the function that relates the signal loss with the distance between transmitter and receiver is a logarithmic function. Hence, the mean $\mu(d_k)$ of the likelihood function (10) can be expressed as shown in (15). The standard deviation of the likelihood function has been assumed to depend linearly with the distance and then can be expressed as shown in (16).

$$\mu(d_k) = A \log(Bd_k) + C \quad (15)$$

$$\sigma(d_k) = Dd_k + E \quad (16)$$

The parameters A , B , C , D and E have been calculated offline from data sets collected during real experiments using a least squares process in MATLAB. The resulting expressions are:

$$\mu(d_k) = 20.2253 \log(20.5411d_k) + 20.9942 \quad (17)$$

$$\sigma(d_k) = -0.0015d_k + 3.0892 \quad (18)$$

Fig. 1 Signal loss as a function of the distance calculated from training data logs collected during flight tests

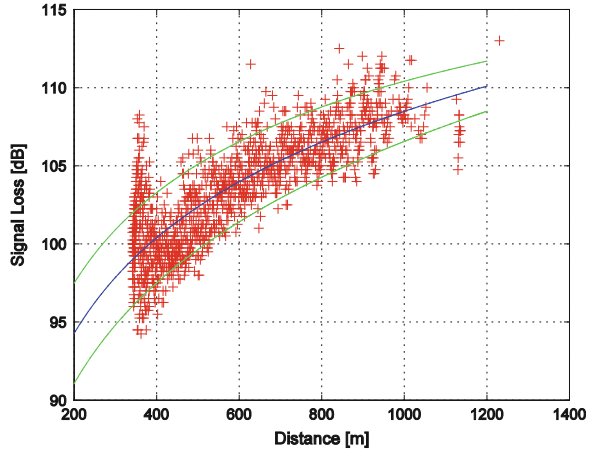


Figure 1 shows the mean function $\mu(d)$ in blue and the functions $\mu(d) + \sigma(d)$ and $\mu(d) - \sigma(d)$ in green. The signal loss measures are also represented versus the distance.

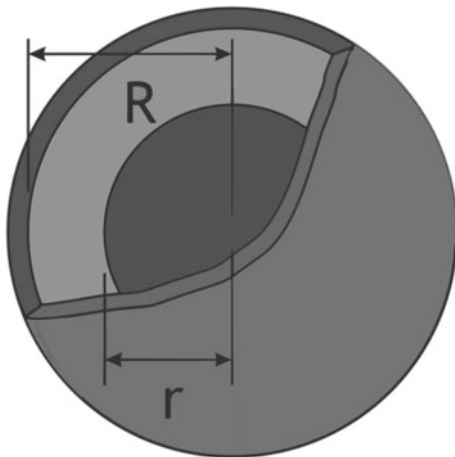
3.2 Particle Filter-Based Localization Algorithm

The initialization of the algorithm consists of setting the initial particles and the associated weights. For this purpose, the first signal loss measures are used to calculate the mean distance between the receiver and the transmitter and its variance using the inverse formulas of (14) and (15). With this information, the particles are set uniformly in a region between two concentric spheres centered at the position of the UAV with radius equal to $r = \mu - \sigma$ and $R = \mu + \sigma$ as shown in Fig. 2. To have a good resolution (i.e., many particles per volume unit) the filter is initialized only when the value $\mu - \sigma$ calculated from the signal loss measures is below a certain threshold (in this work 105 dB has been chosen). This allows the particles to be distributed over a smaller region and with higher density. The initial weights are simply set as $1/N_s$ where N_s is the number of particles in use. In the implementation presented in this paper 4000 particles have been used. This implies that at the beginning all the particles have the same probability.

Each iteration particles are updated following the transition distribution $p(\mathbf{x}_k/\mathbf{x}_{k-1})$. As the transmitter is moving slowly with respect to the UAV a random move is added to the particles in order to search locally over the position space around the position of the previous time step. This is done even if the case where the transmitter is considered to be static [10]. A uniform random movement has been chosen so the prediction stage can be written as

$$p(\mathbf{x}_k/\mathbf{x}_{k-1}) = U(\mathbf{x}_{k-1} - \sigma_{\mathbf{m}}, \mathbf{x}_{k-1} + \sigma_{\mathbf{m}}) \quad (19)$$

Fig. 2 Region where the initial particles are set



where the maximum value of the random movement σ_m has been chosen to be 20 m.

Each time a new signal loss measure is available, the weights of each particle are updated as

$$w_k^i = \frac{1}{\sigma(d_k^i)\sqrt{2\pi}} \exp\left(-\frac{(L_k - \mu(d_k^i))^2}{2\sigma^2(d_k^i)}\right)w_{k-1}^i \tag{20}$$

where the distance associated with each particle is calculated as $d_k^i = \|\mathbf{x}_k^i - \mathbf{x}_k^{UAV}\|$. The new weights have to be normalized as $w_k^i = w_k^i / \sum_i w_k^i, i = 1, \dots, N_s$.

The mean and variance of the particle positions can then be calculated as $\mu_k = \sum_{i=1}^{N_s} w_k^i x_k^i$ and $\sigma_k^2 = \sum_{i=1}^{N_s} w_k^i (x_k^i - \mu_k)^2$. If this variance is below a certain threshold during a certain time the filter has converged and the mean can be adopted as the estimate of the position. In this work, convergence is considered when the variance is below 100 m² for at least 20 consecutive iterations.

Finally, the effective number of particles is calculated using (9) and if it is smaller than the 10 % of the previous number of particles the resampling process is executed.

3.3 Improved Algorithm Using DEM Data

In applications where the WSN nodes to be located are assumed to be on the ground, the height coordinate of all the particles should be restricted to the terrain elevation corresponding to the particle position $[x, y]$. For this purpose, DEM (Digital Elevation Model) data can be used to force the height coordinate of each particle.

The particle-filter based algorithm that was proposed in the previous sections has been modified to include DEM data to improve the estimation accuracy. DLR's

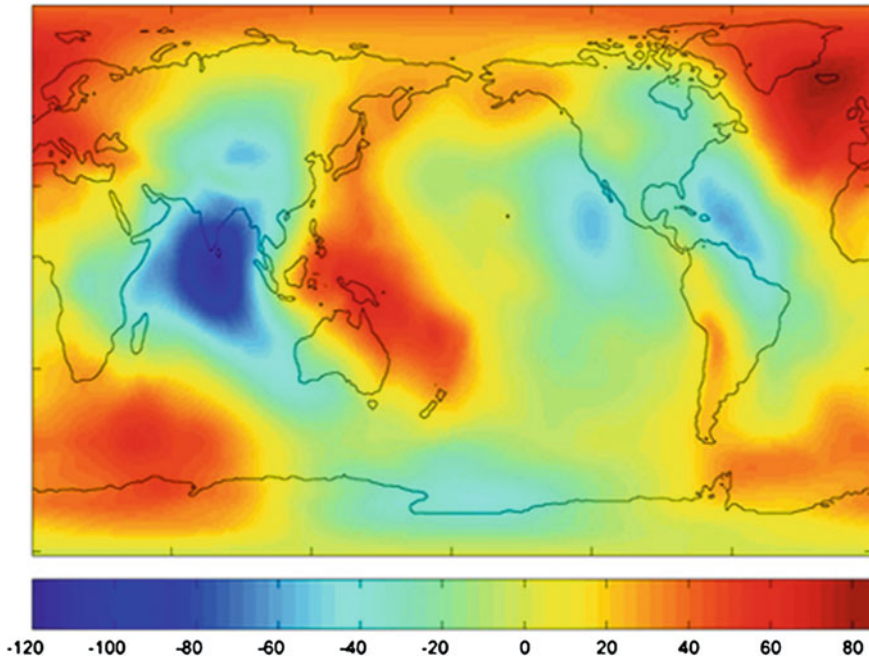


Fig. 3 Worldwide geoid undulation (in meters) [22]

EOWEB (Earth Observation Data Service) service [20] have been used as the source for DEM. These data have been captured during the SRTM (Shuttle Radar Topography Mission) mission of NASA an on-board X-band SAR was used to collect elevation data [21]. SRTM1 files with a tile resolution of 1 arc-second (i.e. about 30m) and a height resolution of 1 m have been used.

It is important to note that elevation data in DEM models are normally referred to the Earth reference geoid. On the other hand, WGS84 (World Geodetic System 84) which is used by the UAV navigation system refers the height data to the Earth reference ellipsoid. Hence, there is a difference in the heights provided by these two systems which is called the geoid undulation (see Fig. 3). This geoid undulation is known for each latitude and longitude and it can be obtained from different online sites (e.g. [23]). In this work WGS84 reference system has been adopted as the common reference system so the elevation obtained from the DEM data has been converted to WGS84 taking the geoid undulation into account.

The particle-based localization algorithm can finally be written as:

```

 $\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s} = \text{PF}[\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k = \{\mathbf{x}_k^{UAV}, L_k\}].$ 
FOR  $i = 1:N_s$  DO
1: Update  $\mathbf{x}_k^i$  according to (18).
2: Compute  $d_k = \|\mathbf{x}_k - \mathbf{x}_k^{UAV}\|$ .
3: Compute  $L_k = P_k^{(t)} - P_k^{(r)}$ .
4: Calculate  $\mu(d_k)$  and  $\sigma(d_k)$  using (16) and (17).
5: Correct the height of the particles with DEM data.
6: Update the weights  $w_k^i$  using (19).
END FOR
7: Normalize the weights:  $w_k^i = w_k^i / \sum_i w_k^i, i = 1, \dots, N_s$ .
8: Calculate  $N_{eff}$ .
IF  $N_{eff} < N_{th}$  THEN
9: Resample.
END IF

```

4 Experiments and Results

To validate and assess the implementation of the localization algorithm, a radio device acting as a receiver was installed on-board a fixed wing UAV and a radio device acting as a transmitter was installed on a 1 m mast on the ground. These devices emulate the beacon node on-board the UAV and the ground-based node to be located respectively in a WSN case. Commercial WiMAX-certified radio equipments from Alberta Systems [24] were selected. WiMAX technology has been used instead of WSN technology to overcome the power limitation of commercial WSN systems, in order to be able to test the algorithm with longer distances between the UAV and the ground transmitter. The localization algorithm has been fully implemented in C++. The embedded PC board was based on an x86 Intel Atom E6x0T processor that runs at 1.6 GHz and included 2 GB of RAM.

Due to the restrictions imposed by current civil legislation to the flights of UAV in non-segregated airspace, the UAV did not move away more than 1.5 km from the security pilot who was located on one side of the runway (so visual line-of-sight conditions were always fulfilled). The flights were performed using an X-Vision UAV manufactured by Elimco (which is shown in Fig. 4) [25]. This UAV flies autonomously using a Piccolo II autopilot from Cloud Cap Technology [26] that provides UAV position data to the embedded PC where the localization algorithm runs. Additionally, the embedded PC receives the values of the transmitted power of the ground station as well as the measured RSSI values from the on-board WiMAX device.

WSNs generally use omnidirectional antennas so these types of antennas were used both on the ground and on-board. This is also beneficial for the localization



Fig. 4 X-Vision UAV used in the experiments



Fig. 5 Flight path followed by the UAV in the experiments. The *green dot* indicates the location of the ground device

algorithm as reduces the effects of antenna gains. A blade antenna was installed on the UAV tail to prevent shadowing from the fuselage, and a monopole antenna was installed on a mast on the ground. To reduce the effects of the antenna patterns in the results, the flight paths were also chosen with long straight paths (as shown in Fig. 5) in order to minimize the turn angles.

All flight experiments were carried out from a rural runway located in Utrera (Seville). The geoid undulation in this area is 48.22 m as obtained from [23].

The experiments were carried out in two stages. In the first one, flights were performed to collect data for calculating the law that relates the signal loss with the distance as discussed in Sect. 3.1. For this purpose, six flights were conducted following trajectories with shapes similar to that shown in Fig. 5 and different altitudes. First of all the position of the ground transmitter was obtained using an RTK GPS receiver with centimeter-level accuracy. During these flights, the on-board PC logged the received signal strength value together with the UAV position and the transmitted power (which can be obtained from the data frames send by the ground transmitter). The UAV position was provided by the UAV navigation system that also uses an RTK GPS receiver so centimeter-level accuracy for the UAV position is also expected. The logged data was then processed offline using a least squares fitting process to get the signal loss versus distance function to be used during operative flights (the curve that was obtained was already shown in Fig. 1).

The second stage of the experiments was conducted some weeks later. As in the first stage of the experiments the flight path was chosen to reduce the effects of the antenna gains in the results. In this case the flight path that was used is shown in Fig. 5 and the UAV flew at a constant altitude of 300m. The position of the ground transmitter was obtained using an RTK GPS receiver to be used as the ground truth (i.e. the reference that is compared with the estimation results). In these flights the on-board PC run the localization algorithm which process the received signal strength value, the UAV position and the transmitted power using the signal loss versus distance function to estimate the transmitter position. The on-board PC also sent the position estimations to a ground PC in real-time during the flight using the WiMAX link. A GIS (Geographical Information System) running in the ground PC is used to represent the estimated position and its uncertainty together with the UAV position (see Fig. 6). The circle that represents the estimation uncertainty includes the



Fig. 6 Ground visualization tool. Particle distribution and UAV position (*left*); map visualization (*right*)

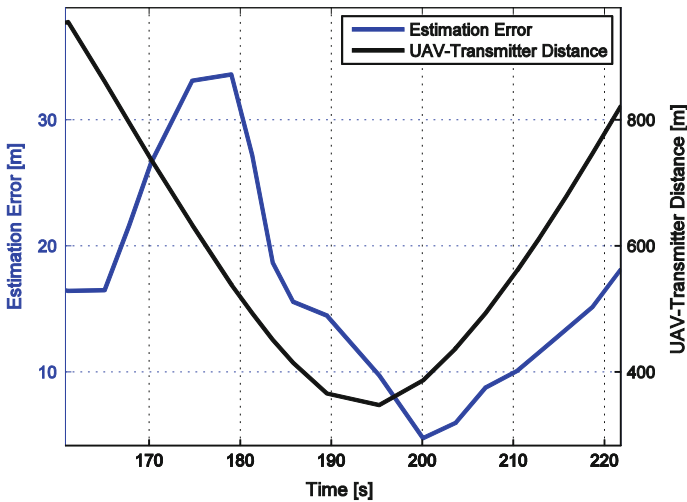


Fig. 7 Localization error versus time experienced in a flight test (the average error is shown in red). Note that the distance between the UAV and the transmitter changes with time according to the flight path

90% of the particles. The distribution of the particles of the filter is also represented for debugging purposes.

Four different flights were conducted in this second stage. The algorithm running on-board has been able to estimate the transmitter location with accuracies better than 120m with a mean value of 60m during the experimental flights. Figure 7 shows the evolution of the localization error with time during one of the flight tests. The estimation error variation with time is due to the movement of the UAV. When the distance between the UAV and the transmitter increases, the localization error also increases. This is due to the fact that when the distance between the UAV and the ground node is large, small variations in signal losses implies large variations in distances as can be seen from Fig. 7. The variation is also due to the relatively high value used for the random movement that is given to the particles. The value used for the random movement allows the algorithm to rapidly adapt the particles to new measurements. This way false convergence states can be avoided at the expenses of experiencing larger variations in the localization error.

5 Summary and Conclusions

Particle Filters estimation methods are approximated implementations of recursive Bayesian estimators. These methods can be used for WSN nodes localization using RSSI measures received by a mobile beacon node. There are many practical cases in which aerial robots are preferred over ground robots to be the mobile beacon

(e.g. with rough terrain or when larger ranges are required). In this chapter the implementation of a Particle Filter-based localization algorithm (originally proposed in [10] and [11]) for WSN node localization from a UAV using RSSI measurements has been presented. The location error of the original algorithm was bigger in the height coordinate than in the horizontal coordinates [10]. To improve this, the original algorithm has been modified to take into account terrain elevation data obtained from DEM files.

While traditionally RSSI-based WSN localization algorithms have been validated and tested in short range scenarios, in the work presented in this chapter has validated the algorithm in real flight tests in which the UAV has flown around with a maximum distance of about 1.5 km to the node to be located. To accomplish this task, the WSN nodes have been substituted by more powerful RF devices (specifically WiMAX radio equipment has been used). The results have shown that the algorithm is able to estimate the 3D position of the ground node with an accuracy better than 120 m when the UAV flies around with a maximum distance to the transmitter of about 1.5 km. These results further extends those presented in [10] where the maximum distance between the UAV and the transmitter was only 80 m.

In the short future the radiation patterns of the antennas will be taken into account in the signal loss calculation. If the antenna pattern is known, the value of the antenna gain to be used can be calculated from the position and attitude of the UAV and the current estimation of the transmitter position. The algorithm will also be tested with a slow moving transmitter (mount on a car moving at low speed).

References

1. Zanca, G., Zorzi, F., Zanella, A., Zorzi, M., Max, M.: Experimental comparison of RSSI-based localization algorithms for indoor wireless sensor networks. In: Workshop on Real-World Wireless Sensor Networks, 2008, pp. 1–5
2. Peng, R., Sichitiu, M.L.: Probabilistic localization for outdoor wireless sensor networks. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **11**(1), 53–64 (2007)
3. Luo, X., O'Brien, W.J., Julien, C.L.: Comparative evaluation of received signal-strength index (RSSI) based indoor localization techniques for construction jobsites. *Adv. Eng. Inform.* **25**(2), 355–363 (2011)
4. Oguejiofor, O.S., Okorogu, V.N., Abe, A., Osuesu, B.O.: Outdoor localization system using RSSI measurement of wireless sensor network. *Int. J. Innov. Technol. Explor. Eng.* **2**(2), 1–6 (2013)
5. Sichitiu, M.L., Ramadurai, V., Peddabachagari, P.: Simple algorithm for outdoor localization of wireless sensor networks with inaccurate range measurements. In: International Conference on Wireless Networks, 2003
6. Benbadis, F., Friedman, T., Dias de Amorim, M., Fdida, S.: GPS-free-free positioning system for wireless sensor networks. In: Second IFIP International Conference on Wireless and Optical Communications Networks, 2005. WOCN 2005, pp. 541–545
7. Morelli, C., Nicoli, M., Rampa, V., Spagnolini, U., Alippi, C.: Particle filters for Rss-based localization in wireless sensor networks: an experimental study. In: 2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings, 2006, Vol. 4, pp. IV-957–IV-960

8. Sichitiu, M.L., Ramadurai, V.: Localization of wireless sensor networks with a mobile beacon. In: 2004 IEEE international conference on mobile Ad-hoc and sensor systems (IEEE Cat. No. 04EX975), pp. 174–183
9. Amundson, I., Koutsoukos, X.D.: A survey on localization for mobile wireless sensor networks. In: 2nd International Conference on Mobile Entity Localization and Tracking in GPS-less Environments (MELT'09), 2009, pp. 235–254
10. Caballero, F., Merino, L., Maza, I., Ollero, A.: A particle filtering method for wireless sensor network localization with an aerial robot beacon. In: 2008 International Conference on Robotics and Automation, pp. 596–601, May 2008
11. Caballero, F., Merino, L., Gil, P., Maza, I., Ollero, A.: A probabilistic framework for entire WSN localization using a Mobile Robot. *Rob. Auton. Syst.* **56**(10), 798–806 (2008)
12. Korner, F., Speck, R., Goktogan, A.H., Sukkarieh, S.: Autonomous airborne wildlife tracking using radio signal strength. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 107–112
13. Posch, A., Sukkarieh, S.: UAV based search for a radio tagged animal using particle filters. In: Australasian conference on Robotics and Automation (ACRA), 2009
14. Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.* **10**(3), 197–208 (2000)
15. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **50**(2), 174–188 (2002)
16. Bergman, N.: Recursive Bayesian estimation: Navigation and Tracking applications, Linköpings University, Sweden (1999)
17. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. Massachusetts Institute of Technology, Cambridge (2005)
18. Bekir, E.: Introduction to Modern Navigation Systems. World Scientific Publishing, Hackensack (2007)
19. Friis, H.T.: Proceedings of the Institute of Radio Engineers, vol. 34, p. 254 (1946)
20. DLR, EOWEB (Earth Observation Data Service). http://eoweb.dlr.de:8080/free_SRTM_X-band_data.html. Accessed 20 Mar 2014
21. DLR, SRTM in DLR Earth Observation Center. http://www.dlr.de/caf/en/desktopdefault.aspx/tabid-5515/9214_read-17716/. Accessed 20 Mar 2014
22. GRACE, Gravity Recovery and Climate Experiment—Earth gravity definition. http://www.csr.utexas.edu/grace/gravity/gravity_definition.html. Accessed 28 Mar 2014
23. UNAVCO, Geoid Height Calculator. http://www.unavco.org/community_science/science-support/geoid/geoid.html. Accessed 20 Mar 2014
24. ARBA MOBILE Product Catalog, Albentia Systems. http://www.albentia.com/Docs/ARBA_Mobile—Catalogo_de_producto_EN.pdf. Accessed 14 Mar 2014
25. UAV X-Vision (E500), Elimco. http://www.elimco.com/p_UAV-E500_25.html. Accessed 14 Mar 2014
26. CloudCapTechnologies, Piccolo II Datasheet. [http://www.cloudcaptech.com/Sales and Marketing Documents/Piccolo II Data Sheet.pdf](http://www.cloudcaptech.com/Sales_and_Marketing_Documents/Piccolo_II_Data_Sheet.pdf). Accessed 14 Mar 2014

Fundamental Limits of Self-localization for Cooperative Robotic Platforms Using Signals of Opportunity

Mei Leng and Wee Peng Tay

Abstract A fundamental problem in robotic applications is the localization of the robots. We consider the problem of global self-localization for a robotic platform with autonomous robots using signals of opportunity (SOOP). We first give a brief overview of the state-of-the-art in robotic localization using SOOP, and then propose a scheme that requires minimal prior environmental information, no pre-configuration, and only loose synchronization between the robots. To further analyze the potential for the use of SOOP in robotic localization and to investigate the effect of clock asynchronism, we derive an analytical expression for the equivalent Fisher information matrix of the Cramér-Rao lower bound (CRLB). The derivation is based on the received signal waveform, and allows us to analyze the contributions of various factors to the localization accuracy. The CRLB provides a valuable guideline for the design of a robotic platform in which a desired level of localization accuracy is to be achieved. We also analyze the distortions in the time difference of arrival and frequency difference of arrival measurements caused by different clock offsets and skews at the robots. We propose a robust algorithm to estimate robot location and velocity, which mitigates the clock biases. Simulation results suggest that our proposed algorithm approaches the CRLB when clock skews have small standard deviations.

1 Introduction

An autonomous robot is a robot capable of intelligent motion and action without human assistance. Equipped with on-board programs, actuators and a wide range of sensors, an autonomous robot has the ability to sense information about its surrounding environment and at the same time navigate itself through its operating environment to perform designated tasks. A robotic platform in this chapter consists

M. Leng (✉) · W.P. Tay
Nanyang Technological University, Singapore, Singapore
e-mail: lengmei@ntu.edu.sg

W.P. Tay
e-mail: wptay@ntu.edu.sg

© Springer International Publishing Switzerland 2015
A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604, DOI 10.1007/978-3-319-18299-5_8

of a team of autonomous robots. Such a robotic platform finds extensive applications in practice, and is especially desirable in environments that are inconvenient or dangerous for humans to work in. For example, a robotic team can assist fire-fighters by navigating through smoke-filled areas, and detecting hidden dangers by gathering relevant environmental information. Robots can also be used to guide people in museums, shopping malls, and office buildings. In industrial environments, the use of multi-robot platforms is heavily linked with navigation and surveillance in various applications such as product tracking in warehouse and safety monitoring at construction sites.

1.1 The Localization Problem

A fundamental problem in robotic applications is the localization of robots in the system [1], where each robot is required to determine its position with respect to a common frame of reference. It is essential to obtain reliable and accurate locations because an erroneous location can lead to hazards such as collisions and performing tasks in the wrong place [2]. A popular technique to localize a robot is Dead Reckoning (DR) [3], which updates the position and heading of a robot by integrating kinetic information, including velocity, acceleration, and time, from the robot's proprioceptive sensors. It requires each robot to be provided with an initial location, and its performance is unreliable due to the fact that errors will accumulate when traveling over long distances or over an uneven surface [4]. A significant body of literature has investigated methods to provide accurate location estimates. Depending on the source of signals used for localization, they can be categorized into three groups: self-initiative signals, dedicated signals, and signals of opportunity (SOOP).

Self-initiative signals are signals generated by exteroceptive sensors that monitor the environment for features. Vision sensors like cameras are typical for service and surveillance robots, and other sensors include sonars, laser scanners, star trackers, and compasses. A common technique used to localize robots with exteroceptive sensors is the landmark method [5], where signals are processed to extract landmarks from its surrounding environment so that the robot is localized relative to landmarks with known positions. Generally, accompanied with prior knowledge of the environment, for example map, images, and laser imaging, the robot can perform a database matching [6] to estimate its own position. These methods are however sensitive to dynamic environments and cannot be applied in uncharted areas.

Dedicated signals are generated specifically for localization. The dominant technology for outdoor applications is the Global Navigation Satellites System (GNSS), and the Global Positioning System (GPS) has been widely employed in many commercial applications. However, outfitting GPS devices on all robots increases system cost and energy drain. More importantly, a GNSS relies on satellite signals that are generally limited to areas with a clear sky view and is vulnerable to disruption. It fails to operate inside most buildings and in "urban canyon" environments. In addition, most works for robot localization has relied on pre-deployed infrastructures. Representative works include the Cricket indoor location system and Cricket-enabled

moving robots [7–9], Active Bat [10], DOLPHIN [11, 12], RADAR [13], and other applications such as the active badge system [14] and Ubiscene [15]. For example, Cricket implements a set of active anchors at known locations. The beacons transmit a combination of RF signal and ultrasonic pulses periodically. The robot passively listens to the beacons and localizes itself by triangulation using time-of-flight information extracted from its received signal. These systems all require the configuration of anchors in advance, and a separate analysis of the operating environment, with tailored solutions for different applications. The high cost of maintenance and the high dependency on environments prevent such systems from being used in remote areas under harsh conditions.

The SOOP are public signals transmitted for various non-localization applications [16], such as AM/FM radio signals, cellular communication signals, digital television signals from satellites and TV towers, and WLAN signals in buildings. The transmitters are called “beacons”, and these signals conform to well-established standards and can be easily detected in most urban areas. As robots move in dense urban areas and deep inside buildings, they can benefit from exploring hybrid signals that provide a good coverage and complement each other in different situations. Robots can be programmed in advance with knowledge of beacon locations and utilize them as anchors for localization. The utilization of public beacons removes the necessity of deploying anchor robots and infrastructure, and hence it is more energy and cost efficient in terms of maintaining the network. In Sect. 1.3, we provide a survey on state-of-the-art techniques on robot localization using SOOP. Each method has advantages and disadvantages that make it more suitable for specific situations.

1.2 Cooperative Localization of Multiple Robots

Many applications require robots collaborate to perform a given task [17]. Robots can communicate with each other to exchange relative position measurements and share environmental information to improve the accuracy of position estimations. The concept of cooperative localization for multiple robots was first proposed in [18], and subsequently many similar robot cooperation schemes have been developed [19–22]. Specifically, the robots are divided into two teams, which move alternatively, with each team serving as an artificial landmark to the other team. Robots in such a system generally are able to roughly locate itself by dead reckoning, and the robot-to-robot measurements are integrated to improve the local estimates. The disadvantages are that only one robot team is allowed to move at any given time, and the two teams must move in such a way that they can “see” each other (visually, electromagnetically, or with sonar) at all times.

In practice, it is desirable that all robots can move simultaneously and randomly without path constraints, and cooperative localization under such conditions has been investigated under probabilistic estimation frameworks. When every robot has access to the map of the environment, a probabilistic method has been studied in [23] to approximate the robot location with a sampling-based representation, and a particle

filter was applied to integrate odometry measurement, environmental measurement and robot-to-robot measurement to achieve global localization. A minimum-entropy approach was explored in [24] for a similar scenario.

To further remove the requirement of using a map, a group of algorithms has focused on decentralized cooperative localization, where each robot can propagate its state and covariance estimates to either a fusion center, all other robots, or its neighboring robots. Various estimation techniques, such as Extended Kalman Filter (EKF) [25–27], Maximum Likelihood Estimation (MLE) [28, 29], and Maximum A Posteriori (MAP) [30, 31], have been applied to process the propagated estimates. The performance upper bound using EKF was analyzed in [32, 33]. It shows that the localization accuracy is related to both measurement noise and the robots' relative geometry. However, in order to achieve global localization, at least one anchor robot must be deployed in such systems.

In this chapter, with the utilization of SOOP, we propose a cooperative localization scheme for robots where no anchor robots are required and no prior maps are necessary. Any two robots can perform self-localization as long as they can detect the same set of SOOP beacons.

1.3 State-of-art Techniques for Robot Localization Using SOOP

There are many SOOP available for navigating robots. For indoor environments, typical SOOP used by robot localization are the RF signals from Wireless Ethernet LAN (WLAN) [34–36], magnetic fields [37], and indoor lights [38]. For urban environments, typical signals include those from mobile base stations and radio stations. Telecommunication signals, including GSM, 3G, and 3GPP LTE [16, 39–41], are widely available in urban environments. AM/FM radio signals [42] also covers a large area in most cities and towns. Compared to GPS signals, they have relatively high power and are able to penetrate walls and buildings. Therefore, robots moving in both city areas and indoors can readily detect signals spreading over a wide frequency range, which provides good signal diversity. For urban areas with partial sky view and remote rural areas, all satellites, including communication satellites, weather satellites, and digital TV satellites, are potential SOOP beacons.

Despite the different characteristics of various SOOP, most techniques for robot localization are based on measurements extracted from these signals. The most widely used measurement for robot localization is received signal strength (RSS). Depending on whether a specific signal propagation model has been assumed, there are several kinds of algorithms in the literature. A direct method is to assume that a known function, such as the Friis equation for RF signals, exists to describe the decaying rate of the signal power or intensity with respect to distance [43], and the RSS values at a robot location can be converted to distance with respect to the beacon. After collecting RSS values with respect to multiple beacons, the robot location can

be estimated by triangulation [44]. This method is simple and cost-effective, but the main drawback is that RSS values have a large variation due to the deleterious effects of fading, shadowing, or non-Gaussian interference. The dependence between RSS values and distance is often complicated and unpredictable in urban environments, and no simple function can describe the propagation accurately.

Many works hence focused on approximating the signal propagation model in a probabilistic way. An early work in [39] models the distribution of RSS for each cellular base station using the Gaussian process, and the location of a robot is estimated by maximizing a joint likelihood function of the distributions of the nearby beacons. Similar works have been done in [34, 35, 45] using WiFi signals. These methods involve a time-consuming calibration procedure where RSS data must be collected at a set of predefined checkpoints spread over the environment so that the RSS distribution conditioned on locations can be constructed via either sampling or curve-fitting. To remove the reliance on the calibration data with location labels, a Gaussian process [46] was developed to model the RSS distribution for sparse training data, and the Gaussian process latent variable models (GP-LVM) [47] were later proposed for on-site training at random unknown locations. Recently, the work in [48] has proposed to learn the relationship between the geographic distance and the RSS measurements by exploring the correlation structure in the spatio-temporal domains. This method does not require checkpoint locations, but it requires extra beacon-to-beacon measurements to build up the signal-distance map for the operating environment, which involves modifying the beacon settings and cannot be easily applied in many applications.

Another typical approach that has been widely investigated for robot localization is the fingerprint method [37, 38, 42, 49]. It also requires a calibration phase where a set of RSS measurements are observed at various location and a database known as a “radio map” is built based on these measurements and the ground-truth locations [50]. The fingerprint method assumes that similar signal strength fingerprints must correspond to a similar location on the map, which however does not hold for all situations. Moreover, due to the substantial cost incurred by to perform calibration, the fingerprint method may be prohibitive for many applications.

Other measurements include angle-of-arrival (AOA), time-difference-of-arrival (TDOA), and frequency-difference-of-arrival (FDOA). Localization based on AOA requires complex antenna arrays for angle measurement, and the position accuracy deteriorates as the distance to the beacon increases, which prohibits its use on robots working in remote areas or in harsh conditions. On the other hand, the major challenge for using SOOP lies in the fact that beacons are non-cooperative and robots typically have no knowledge of important information like beacon transmit power, transmit time, and signal waveforms. TDOA and FDOA provides a practical solution [40], since they can remove the ambiguity resulting from unknown transmit times or unknown signal waveforms. However, to the best of our knowledge, very few works has been done for robot localization using TDOA and FDOA. Its development has been hampered by several hurdles including multi-path and non-line-of-sight errors for RF signals, and the need for high sampling rates to achieve better time resolution.

In this chapter, we discuss one of the most critical challenges faced by TDOA and FDOA methods in the use of SOOP for robotic localization, which is the issue of time synchronization between robots and beacons. In order to obtain reliable timing information, it is essential that robots and beacons are strictly synchronized, and most existing methods make this simplifying assumption. However, clock synchronization is difficult to achieve and maintain in practice [51], and most beacons, like cellular base stations and WiFi access points, are unaware of the presence of the robots and will not actively synchronize with them or a common universal clock. Unlike GPS signals that are generated by satellite atomic oscillators with extremely small skews (less than 10^{-11}), SOOP beacons are usually equipped with less perfect oscillators that have clock skews varying from 10^{-8} to 10^{-4} , which result in an accumulated clock offset up to 0.1 ms for 1 s.

1.4 Contributions of the Book Chapter

In this chapter, we focus on the problem of global self-localization for a robotic platform with autonomous robots. We assume that robots perform localization using only SOOP, and they have access only to minimal a priori environmental information. We also assume that SOOP beacons and robots are not fully synchronized. We propose a cooperative localization scheme for robots using SOOP. This scheme allows robots to move randomly and simultaneously without any pre-configuration. We investigate the performance bound for robot localization, and we focus on the fundamental limits for robot location and velocity estimation using asynchronous beacons. The analytical expression of the equivalent Fisher information matrix (EFIM) enables us to decompose the accuracy uncertainties into three components associated with beacon geometry, signal characteristics and clock skew, respectively. It clearly demonstrates how different components affect the localization performance and gives us various insights for designing practical robot localization methods. We show that the approximate EFIM does not depend on robot and beacon clock offsets, which suggests that there exists estimation procedures that does not require a priori knowledge of these quantities.

2 Cooperative Localization Using SOOP

2.1 System Model Based on Two Robots

As two robots move in an operating environment, the type of SOOP that the robots can receive depends on where the robots are. Two example scenarios are shown in Fig. 1. For typical urban outdoors, satellites, radio stations, and cellular base stations provide a wide range of RF signals as SOOP. For indoor environments such as office buildings

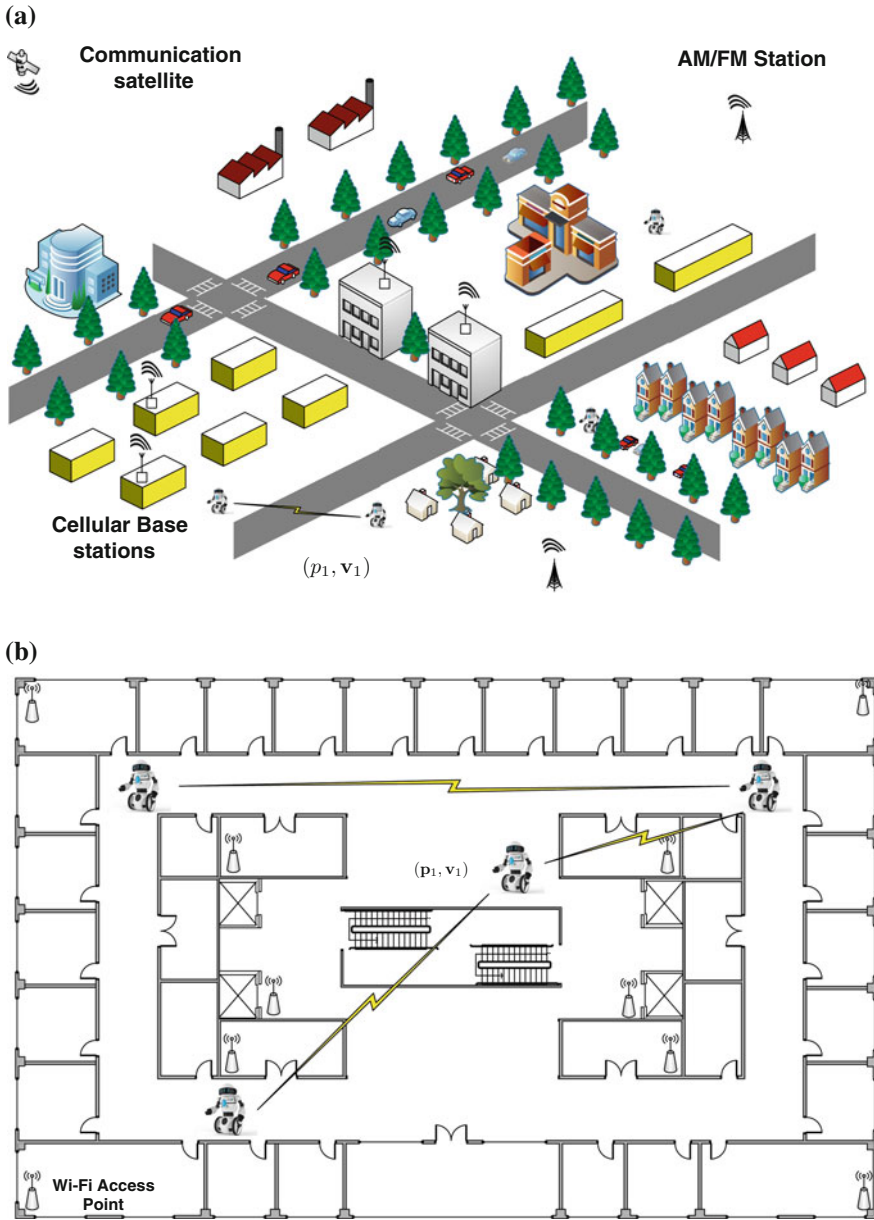


Fig. 1 Different types of SOOP beacons are available in different environments. **a** Outdoor. **b** Indoor

and factories, WiFi and Bluetooth signals are widely available. Without placing stringent constraints on the SOOP, we only require that the two robots can receive signals from the same beacon during the same observation period. We consider a general scenario where Assumption 1 holds.

Assumption 1

- (i) The robots can differentiate signals from different beacons, and have prior knowledge of the beacon locations, velocity (if the beacon is moving), and nominal transmit frequency.
- (ii) The robots can communicate with each other. Specifically, they can share information via handshaking, and can exchange locally received signal segments.

Assumption 1 (i) can be satisfied by allowing the robots to first learn about the environment they are operating in. For signals transmitted in the same frequency band, additional information may be provided. For example, PN codes for systems using CDMA can be programmed in advance in the robots. Other characteristics such as signal bandwidth and modulation schemes may also be known. In general, robots can scan over a wide spectrum so that different signals can be detected. The beacons whose information is available at robots are called “registered beacons”. When two robots are deployed, the following cooperation scheme is carried out to perform self-localization. In Sect. 2.2, we discuss extensions of our proposed scheme to a team of more than two robots.

1. Each robot searches over the spectrum for available SOOP from registered beacons. Once it succeeds, the robot sends the registered beacon ID to its fellow robot. When two robots detect the same beacon, they will make an agreement on when to start receiving the SOOP via a handshaking procedure. Non-registered beacons cannot be used by robots, since their information, such as carrier frequency and locations, are unknown.
2. After receiving the acknowledgment of the receiving time and the receiving duration, each robot starts to record a short signal segment, and the two robots exchange their received signals over a wireless channel.
3. Each robot performs a cross-correlation using two received signals and estimate the corresponding TDOA and FDOA measurements.
4. When multiple pairs of TDOA and FDOA measurements are available, each robot can localize itself using the differential TDOA (DTDOA) estimator and velocity estimation method discussed in Sect. 3.2, where each pair of measurements is with respect to one registered beacon that has been detected by both robots.

The cooperative scheme above describes a general procedure for making use of SOOP. It can also be applied between a robot and a pre-deployed sensor, where the sensor acts as a relay station for SOOP and assists the robot to perform its self-localization. Without loss of generality, we limit our discussion to the cooperation between robots in the following. The two robots do not need to know the signal waveforms or the transmit times from the SOOP beacons. As stated in Sect. 1.3, one critical issue for using SOOP is time synchronization. In our proposed cooperative

scheme, we do not require strict synchronization among robots and beacons. In order to guarantee that two robots receive from the same beacon within an overlapping observation window, it is sufficient for the receive time to be scheduled with a synchronization error on the order of a second. Therefore, we allow robots to have loose synchronization between each other such that a small clock offset exists before receiving the SOOP. The loose synchronization can be easily achieved by calibration before the deployment of the robots. After deployment, each robot's local clock also drifts with a constant rate during the observation period. It is expected that such clock asynchronism will introduce distortions into the TDOA and FDOA measurement in Step 3, and we propose a location and velocity estimator in Sect. 3.2 to handle such errors.

Since beacon locations are generally with respect to the global coordinate frame, the proposed scheme can achieve absolute localization with SOOP, and does not require a reference node in the team, in contrast to the case of relative localization using robot-to-robot measurements. The utilization of SOOP beacons also removes the necessity of deploying an artificial landmark or an anchor robot as in most existing cooperative schemes. Therefore, a team of autonomous robots can achieve self-localization without any pre-configuration or deployment of special infrastructure, and this enables the robotic platform to work in remote areas, and under harsh conditions.

2.2 Extension to Multiple Robots

The cooperative localization scheme we propose can be easily generalized to a team of more than two robots through the following natural extensions. We randomly choose a robot, and let it broadcast its received signal from each beacon to all the other robots in the network. Each robot can then follow the cooperative scheme to estimate both its own location and velocity as well as the broadcasting robot's location and velocity. Alternatively, a distributed procedure can be implemented in which robots exchange signals with each other only if they are within a certain range of each other. In this case, each robot may have access to information from multiple neighbors, from which TDOA and FDOA measurements can be computed. A distributed estimation procedure based on [52] can then be implemented to determine all the robots' location and velocities iteratively.

3 Fundamental Limits Under Clock Asynchronism

Since we do not require stringent synchronization on the beacons and the robots, it is important to analyze how their clock asynchronism will affect the localization performance. We suppose that each beacon b has a local oscillator operating with

clock skew β_b and clock offset Ω_b , so that its local time $t_b(t)$ with respect to a universal standard time t is given by [51]

$$t_b(t) = \beta_b t + \Omega_b. \quad (1)$$

Similarly, each robot j has a local oscillator with local time given by $t_j(t) = \beta_j t + \Omega_j$. When two robots are deployed, their clocks are calibrated with a loose synchronization, and a clock offset at second level may exist. The clock skew characterizes the rate at which an oscillator drifts, and it depends on various random quantities like its quality, power level, and other environmental variables.

We assume that a set \mathcal{B} of N beacons are observed by two robots. The beacon $b \in \mathcal{B}$ has a known fixed position \mathbf{p}_b , and it broadcasts a narrowband signal at a nominal carrier frequency f_b . Two robots, S_1 and S_2 are at unknown locations \mathbf{p}_1 and \mathbf{p}_2 , and moving with unknown velocities \mathbf{v}_1 and \mathbf{v}_2 respectively. Two robots communicate with each other via a wireless channel at nominal frequency f_0 . Denote the set of robots as $\mathcal{S} = \{1, 2\}$. The robot localization problem involves estimating locations \mathbf{p}_j and velocities \mathbf{v}_j for $j \in \mathcal{S}$, using SOOP from beacons. In this section, we first analyze how received signals at each robot are distorted due to clock asynchronous, and then we propose a location and velocity estimator to eliminate the bias terms, and finally we derive the modified CRLB for the estimation performance based on the received signals.

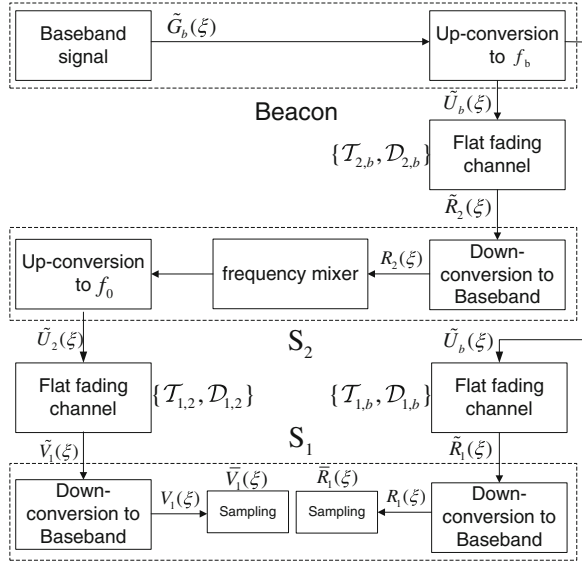
3.1 Distorted Received Signals

Consider the cooperative scheme between two robots, $\{S_1, S_2\}$, and one beacon b . Suppose that beacon b generates a nominal baseband signal $g_b(t)$. Because of clock skew and offset at the beacon, the actual baseband signal is $\tilde{g}_b(t) = g_b(\beta_b t + \Omega_b)$, which is then up-converted to the nominal passband frequency f_b at the beacon for transmission, and the actual passband frequency may differ from f_b due to the clock skew of beacon b . The robot S_1 receives the signal from beacon b , and it performs down-conversion and sampling before it exchange the signal with its fellow robot S_2 . The procedure is further illustrated in Fig. 2. It is obvious that each step involves signal processing using the local oscillator, and it introduces errors due to its clock skew and offset. Specifically, we derive the analytical expression for the received signal at robots, and it is given in Proposition 1, whose proof consists mainly of algebraic manipulations, and can be found in [53].

Proposition 1 *Let $g_b(t)$ be the nominal baseband signal to be generated by beacon b , and $G_b(\xi)$ be its Fourier transform. The received signal $r_{1,b}(t)$ at robot S_1 from beacon b has baseband representation*

$$R_{1,b}(\xi) \propto G_b \left[(\beta_1 \xi - \Upsilon_{1,b}) / (\beta_b \gamma_{1,b}) \right] e^{-i2\pi \xi \frac{\beta_1 \delta_{1,b}}{\beta_b \gamma_{1,b}}}, \quad (2)$$

Fig. 2 Block diagram for communications between beacons and robots



where $\delta_{1,b} = \mathcal{T}_{1,b}\beta_b - \Omega_1\beta_b/\beta_1 - \Omega_b$ and $\Upsilon_{1,b} = f_b(\gamma_{1,b}\beta_b - \beta_1)$. The received signal $v_{1,b}(t)$ at robot S_1 from robot S_2 has baseband representation

$$V_{1,b}(\xi) \propto G_b [(\beta_1\xi - \Psi_{1,b})/(\beta_b\gamma_{1,2}\gamma_{2,b})] e^{-i2\pi\xi \frac{\beta_1\Lambda_{1,b}}{\beta_b\gamma_{1,2}\gamma_{2,b}}}, \quad (3)$$

where $\Lambda_{1,b} = \mathcal{T}_{2,b}\beta_b + \mathcal{T}_{1,2}\gamma_{2,b}\beta_b - \Omega_2\beta_b/\beta_2 - \Omega_b$, and $\Psi_{1,b} = f_b\gamma_{1,2}(\gamma_{2,b}\beta_b - \beta_2) + f_0(\gamma_{1,2}\beta_2 - \beta_1)$.

where $\mathcal{T}_{j,b} = \|\mathbf{p}_j - \mathbf{p}_b\|/c$ denotes the propagation delay between b and S_j with c being the speed of light, $\mathcal{D}_{j,b} = -f_b\mathbf{v}_j^T(\mathbf{p}_j - \mathbf{p}_b)/(c\|\mathbf{p}_j - \mathbf{p}_b\|)$ denotes the nominal Doppler shift, and $\gamma_{j,b} = 1 + \mathcal{D}_{j,b}/f_b$. To simplify the derivation, we have also assumed that every wireless channel has a flat fading time-varying impulse response.

3.2 Robust Algorithm for Location and Velocity Estimation

By cross-correlating the received signals $r_{1,b}(t)$ and $v_{1,b}(t)$, the robot S_1 can obtain its TDOA and FDOA measurement with respect to beacon b as,

$$\hat{\tau}_b \approx \beta_1\mathcal{T}_{2,b} - \beta_1\mathcal{T}_{1,b} + \beta_1\mathcal{T}_{1,2} + \Omega_1 - \frac{\beta_1}{\beta_2}\Omega_2, \quad (4)$$

$$\hat{\xi}_b \approx \frac{\beta_b}{\beta_1}\mathcal{D}_{2,b} - \frac{\beta_b}{\beta_1}\mathcal{D}_{1,b} + \left[\frac{\beta_2}{\beta_1} + \frac{f_b}{f_0}\left(\frac{\beta_b}{\beta_1} - \frac{\beta_2}{\beta_1}\right)\right]\mathcal{D}_{1,2} + \left(1 - \frac{\beta_2}{\beta_1}\right)(f_b - f_0). \quad (5)$$

The approximation is due to the fact that $\gamma_{j,b} = 1 + \mathcal{D}_{j,b}/f_b \approx 1$, since even in an extreme case with a relative speed at the order of km/s, the ratio $\mathcal{D}_{j,b}/f_b$ has a very small value, for example 10^{-6} for a nominal frequency f_b at the order of MHz, and hence it can be safely ignored. It can be seen that the bias terms $\Omega_1 - \Omega_2\beta_1/\beta_2$ and $(1 - \beta_2/\beta_1)(f_b - f_0)$ are included in the estimated TDOA and FDOA respectively, leading to large errors if these estimates are used directly in current TDOA and FDOA localization procedures. For example, typical crystal oscillators have a clock skew of 10^{-8} and can result in a bias of up to 10 ns in the TDOA estimate over a one second interval, and for every 100 MHz frequency difference between f_b and f_0 , a bias of 1 Hz will be introduced in the FDOA estimate. To alleviate the performance degradation caused by clock biases, We in the following propose an algorithm to estimate robot locations and velocities using TDOA and FDOA measurements.

3.2.1 Location Estimation

Let $\beta_j = 1 + \delta\beta_j$ with $\delta\beta_j$ being a random error such that $\mathbb{E}\{\delta\beta_j\} = 0$ and $\mathbb{E}\{\delta\beta_j^2\} = \sigma_{\beta_j}^2$. Substituting into (4), we have

$$c\hat{\tau}_b \approx \|\mathbf{p}_2 - \mathbf{p}_b\| - \|\mathbf{p}_1 - \mathbf{p}_b\| + c(\mathcal{T}_{1,2} + \Omega_1 - \Omega_2) + c(\delta_b^\tau + e_b^\tau), \quad (6)$$

where $\delta_b^\tau = \delta\beta_1(\|\mathbf{p}_2 - \mathbf{p}_b\| - \|\mathbf{p}_1 - \mathbf{p}_b\| + \|\mathbf{p}_1 - \mathbf{p}_2\|)/c - \Omega_2(1 + \delta\beta_1)/(1 + \delta\beta_2)$. Since delay estimations from all beacons, i.e., $\{\hat{\tau}_b\}_{b \in \mathcal{B}}$, experience the same amount of biases caused by propagation delay and robot clock offsets, we subtract two delay estimations and obtain

$$c(\hat{\tau}_b - \hat{\tau}_a) \approx \underbrace{(\|\mathbf{p}_2 - \mathbf{p}_b\| - \|\mathbf{p}_2 - \mathbf{p}_a\|) - (\|\mathbf{p}_1 - \mathbf{p}_b\| - \|\mathbf{p}_1 - \mathbf{p}_a\|)}_{\triangleq \mathbf{f}_{b,a}(\mathbf{x})} + c(\delta_b^\tau - \delta_a^\tau + e_b^\tau - e_a^\tau), \quad (7)$$

for $a \neq b$ and $\mathbf{x} = [\mathbf{p}_1^T, \mathbf{p}_2^T]^T$. We call the quantity $c(\hat{\tau}_b - \hat{\tau}_a)$ obtained in (7) the *differential TDOA* or *DTDOA*. Notice that the noise term $\delta_b^\tau - \delta_a^\tau = \delta\beta_1(\|\mathbf{p}_2 - \mathbf{p}_b\| - \|\mathbf{p}_1 - \mathbf{p}_b\| - \|\mathbf{p}_2 - \mathbf{p}_a\| + \|\mathbf{p}_1 - \mathbf{p}_a\|)/c$, and it depends on the location parameters due to $\delta\beta_1$. However, if σ_{β_j} is small, its effect on the location estimation is negligible. Suppose there exist N beacons, multiple DTDOA can be obtained for $a, b \in \{1, \dots, N\}$. Stacking all available DTDOA into vectors, we have

$$\mathbf{z} = \mathbf{f}(\mathbf{x}) + \mathbf{e}, \quad (8)$$

where $\mathbf{f}(\mathbf{x}) = [\dots, \mathbf{f}_{b,a}(\mathbf{x}), \dots]^T$, similarly $\mathbf{z} = [\dots, c(\hat{\tau}_b - \hat{\tau}_a), \dots]^T$ and $\mathbf{e} = [\dots, c(\delta_b^\tau - \delta_a^\tau + e_b^\tau - e_a^\tau), \dots]^T$ with covariance matrix $\mathbf{Q} = \mathbb{E}\{\mathbf{e}\mathbf{e}^T\}$. We estimate the robot locations by minimizing the least square (LS) error criterion,

$$(\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2) = \arg \min_{\mathbf{x}} (\mathbf{z} - \mathbf{f}(\mathbf{x}))^T \mathbf{Q}^{-1} (\mathbf{z} - \mathbf{f}(\mathbf{x})). \quad (9)$$

A closed-form solution of (9) does not exist in general due to its non-linearity nature. Various algorithms have been investigated in the literature [54–56]. Generally, we can linearize $\mathbf{f}(\mathbf{x})$ by a Taylor series expansion around an initial guess $\hat{\mathbf{x}}_0$ for the true parameter vector, and we have

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\hat{\mathbf{x}}_0) + \mathbf{G}(\hat{\mathbf{x}}_0)(\mathbf{x} - \hat{\mathbf{x}}_0), \quad (10)$$

where $\mathbf{G}(\cdot) = [\dots, \mathbf{G}_{b,a}^T(\cdot), \dots]^T$ with $\mathbf{G}_{b,a}(\hat{\mathbf{x}}_0) \triangleq \nabla_{\mathbf{x}} \mathbf{f}_{b,a}(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}_0}$ is the gradient of $\mathbf{f}_{b,a}(\cdot)$ evaluated at $\mathbf{x} = \hat{\mathbf{x}}_0$ and can be shown as

$$\mathbf{G}_{b,a}(\hat{\mathbf{x}}_0) = \begin{bmatrix} -\frac{\mathbf{p}_1 - \mathbf{p}_b}{\|\mathbf{p}_1 - \mathbf{p}_b\|} + \frac{\mathbf{p}_1 - \mathbf{p}_a}{\|\mathbf{p}_1 - \mathbf{p}_a\|}, \\ \frac{\mathbf{p}_2 - \mathbf{p}_b}{\|\mathbf{p}_2 - \mathbf{p}_b\|} - \frac{\mathbf{p}_2 - \mathbf{p}_a}{\|\mathbf{p}_2 - \mathbf{p}_a\|} \end{bmatrix}_{\mathbf{x}=\hat{\mathbf{x}}_0}^T.$$

An iterative solution of (9) by using gradient descent method is then given by

$$\hat{\mathbf{x}}^{(l+1)} = \hat{\mathbf{x}}^{(l)} + \alpha_l \left(\mathbf{G}^T(\hat{\mathbf{x}}^{(l)}) \mathbf{Q}^{-1} \mathbf{G}(\hat{\mathbf{x}}^{(l)}) \right)^{-1} \mathbf{G}^T(\hat{\mathbf{x}}^{(l)}) \mathbf{Q}^{-1} (\mathbf{z} - \mathbf{f}(\hat{\mathbf{x}}^{(l)})), \quad (11)$$

where α_l is the step size in the l th iteration.

3.2.2 Velocity Estimation

We utilize FDOA estimates from all beacons, i.e., $\{\hat{\xi}_b\}_{b \in \mathcal{B}}$ to estimate the velocities of both robots. From (5) and $\beta_m = 1 + \delta\beta_m$ for $m \in \mathcal{S} \cup \mathcal{B}$, we have

$$\hat{\xi}_b \approx \frac{f_b \hat{\mathbf{u}}_{1,b}^T - f_0 \hat{\mathbf{u}}_{1,2}^T}{c} \mathbf{v}_1 - \frac{f_b \hat{\mathbf{u}}_{2,b}^T - f_0 \hat{\mathbf{u}}_{1,2}^T}{c} \mathbf{v}_2 + \left(1 - \frac{\beta_2}{\beta_1}\right) (f_b - f_0) + (\delta_b^\xi + e_b^\xi), \quad (12)$$

where $\hat{\mathbf{u}}_{2,b} = (\hat{\mathbf{p}}_2 - \mathbf{p}_b) / \|\hat{\mathbf{p}}_2 - \mathbf{p}_b\|$, $\hat{\mathbf{u}}_{2,a}$, $\hat{\mathbf{u}}_{1,a}$ and $\hat{\mathbf{u}}_{1,b}$ are defined similarly, and $\delta_b^\xi = (\delta\beta_b - \delta\beta_1 - \delta\beta_b \delta\beta_1) [(\hat{\mathbf{u}}_{1,b} - \hat{\mathbf{u}}_{1,2})^T \mathbf{v}_1 - (\hat{\mathbf{u}}_{2,b} - \hat{\mathbf{u}}_{1,2})^T \mathbf{v}_2] f_b / c + (\delta\beta_2 - \delta\beta_1 - \delta\beta_2 \delta\beta_1) \hat{\mathbf{u}}_{1,2}^T (\mathbf{v}_1 - \mathbf{v}_2) (f_b - f_0) / c$, and we have approximated $1/(1 + \delta\beta_m) \approx 1 - \delta\beta_m$ for small $\delta\beta_m$. The random variable δ_b^ξ has zero mean and depends on the velocity and the standard deviation of clock skews. When σ_{β_m} is small, its effect is negligible, and we approximate $\delta_b^\xi \approx 0$. Stacking FDOA measurements from all beacons into a vector, we have

$$\underbrace{\begin{bmatrix} \hat{\xi}_1 \\ \vdots \\ \hat{\xi}_N \end{bmatrix}}_{\xi} \approx \underbrace{\begin{bmatrix} (f_1 \hat{\mathbf{u}}_{1,1}^T - f_0 \hat{\mathbf{u}}_{1,2}^T) / c & -(f_1 \hat{\mathbf{u}}_{2,1}^T - f_0 \hat{\mathbf{u}}_{1,2}^T) / c & (f_1 - f_0) / c \\ \vdots & \vdots & \vdots \\ (f_N \hat{\mathbf{u}}_{1,N}^T - f_0 \hat{\mathbf{u}}_{1,2}^T) / c & -(f_N \hat{\mathbf{u}}_{2,N}^T - f_0 \hat{\mathbf{u}}_{1,2}^T) / c & (f_N - f_0) / c \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \frac{c(\beta_1 - \beta_2)}{\beta_1} \end{bmatrix}}_{\mathbf{y}} + \begin{bmatrix} e_1^\xi \\ \vdots \\ e_N^\xi \end{bmatrix} \quad (13)$$

Since the noise term in (13) has zero mean and a variance \mathbf{C} , which can be inferred using the standard performance limit for time delay and frequency delay estimation [57], the robot velocities can therefore be easily estimated using the best linear unbiased estimator (BLUE) [58] and we have $\hat{\mathbf{y}} = (\mathbf{U}^T \mathbf{C}^{-1} \mathbf{U})^{-1} \mathbf{U}^T \mathbf{C}^{-1} \boldsymbol{\xi}$.

3.3 Performance Limit Using Received Signals (2) and (3)

In this section, we derive the fundamental limits for location and velocity estimation using the received signals at S_1 . Let T be the sampling interval, and T_{ob} be the total observation time. For each beacon $b \in \mathcal{B}$, let $r_b[1 : T_{ob}/T]$ and $v_b[1 : T_{ob}/T]$ be the sampled sequence of the received signal from b received at robot S_1 , and the signal from b retransmitted from S_2 to S_1 , respectively. Taking the inverse Fourier transform of (2) and (3), we have for each $l = 1, \dots, T_{ob}/T$,

$$r_b[l] = g_b(\beta_b \gamma_{1,b} l T / \beta_1 - \delta_{1,b}) e^{-i2\pi \gamma_{1,b} l T / \beta_1} + \varpi_b^r[l], \quad (14)$$

$$v_b[l] = g_b(\beta_b \gamma_{1,2} \gamma_{2,b} l T / \beta_1 - \Lambda_{1,b}) e^{-i2\pi \Psi_{1,b} l T / \beta_1} + \varpi_b^v[l]. \quad (15)$$

The terms $\varpi_b^r[l]$ and $\varpi_b^v[l]$ in (14) and (15) are additive complex white Gaussian noises with variance \mathbf{P}_0 . Let $\mathbf{r} = \{r_b[1 : T_{ob}/T] : b \in \mathcal{B}\}$ and $\mathbf{v} = \{v_b[1 : T_{ob}/T] : b \in \mathcal{B}\}$ be the collection of observations from all beacons.

Our analysis is based on the received sequences given by (14) and (15). Treating the robot clock skews as nuisance parameters, there are $4L + 2 + 2N$ unknown parameters $\mathbf{p}_1, \mathbf{p}_2, \mathbf{v}_1, \mathbf{v}_2, \Omega_1, \Omega_2$ and $\{\beta_b, \Omega_b\}_{b \in \mathcal{B}}$, where L is the length of the position vector \mathbf{p}_1 and N is the number of detected beacons. We stack the unknown parameters into a vector and denote it as $\mathbf{x} = [\mathbf{p}_1^T, \mathbf{p}_2^T, \mathbf{v}_1^T, \mathbf{v}_2^T, \Omega_1, \Omega_2, \{\beta_b, \Omega_b\}_{b \in \mathcal{B}}]^T$. Let $\hat{\mathbf{x}}$ be an estimate of \mathbf{x} . To simplify the computations, we use the modified Bayesian CRLB [59–61], which allows us to first treat the robot clock skews β_1 and β_2 as known values, and then taking expectation over all random clock skews. The actual Bayesian CRLB gives a tighter error bound, but has a much more complicated form that unfortunately does not provide additional insights compared to the analysis in this paper. We therefore choose to present the modified Bayesian CRLB instead. We have $\mathbb{E} \{(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T\} \geq \mathbf{F}_x^{-1}$, where \mathbf{F}_x is the Fisher information matrix (FIM), which can be shown to be ¹

$$\mathbf{F}_x = \mathbb{E}_{\boldsymbol{\beta}} \left\{ \mathbb{E}_{\mathbf{r}, \mathbf{v} | \mathbf{x}} \left\{ -\frac{\partial^2 \ln p(\mathbf{r}, \mathbf{v} | \mathbf{x}, \beta_1, \beta_2)}{\partial \mathbf{x} \partial \mathbf{x}^T} \right\} \right\}, \quad (16)$$

where $\boldsymbol{\beta} = \{\beta_m\}_{m \in \mathcal{S} \cup \mathcal{B}}$ are the random clock skews. Since we are interested in estimation accuracies for robot locations and velocities, i.e., $\{\mathbf{p}_j, \mathbf{v}_j\}_{j=1,2}$, it is sufficient

¹The notation $\mathbb{E}_{y|x}$ means taking the expectation over y conditioned on x , while $p(y|x)$ is the probability density function of y conditioned on x .

to find its equivalent Fisher information matrix (EFIM) $\mathbf{F}_{e,s}$ [59]. It follows by partitioning $\mathbf{F}_{\mathbf{x}}$ into four blocks as

$$\mathbf{F}_{\mathbf{x}} = \begin{bmatrix} \mathbf{F}_0 & \mathbf{F}_1 \\ \mathbf{F}_1^T & \mathbf{F}_2 \end{bmatrix},$$

where \mathbf{F}_0 is a $4L$ -by- $4L$ matrix corresponding to the parameter of interests $\{\mathbf{p}_1^T, \mathbf{p}_2^T, \mathbf{v}_1^T, \mathbf{v}_2^T\}$. The EFIM is then given by

$$\mathbf{F}_{e,s} = \mathbf{F}_0 - \mathbf{F}_1 \mathbf{F}_2^{-1} \mathbf{F}_1^T. \quad (17)$$

To facilitate further analysis of the EFIM in (17), we define the signal energy \mathbf{P}_b , the root-mean-square (RMS) bandwidth \mathbf{W}_b , and the RMS integration time \mathbf{T}_b for the signal $g_b(t)$ as [62],

$$\mathbf{P}_b = \int |g_b(t)|^2 dt, \quad \mathbf{W}_b = \left[\frac{\int |f G_b(f)|^2 df}{\int |G_b(f)|^2 df} \right]^{\frac{1}{2}}, \quad \mathbf{T}_b = \left[\frac{\int |t g_b(t)|^2 dt}{\int |g_b(t)|^2 dt} \right]^{\frac{1}{2}}.$$

Without loss of generality, we assume that the signal $g_b(t)$ has zero centroid in time and frequency, hence \mathbf{W}_b and \mathbf{T}_b characterize the signal's energy dispersion around its centroid in time and frequency, respectively. In the following, we make various assumptions and approximations, which hold in most practical applications. We use $a \ll b$ to mean that a/b can be approximated by 0.

Assumption 2

- (i) The clock skew standard deviations $\sigma_{\beta_m} < 1$ for all $m \in \mathcal{S} \cup \mathcal{B}$.
- (ii) For every beacon $b \in \mathcal{B}$, the RMS bandwidth \mathbf{W}_b is much smaller than the nominal carrier frequency f_b with $\mathbf{W}_b \ll f_b$.
- (iii) There exists $\epsilon > 0$ and a measurable set with probability at least $1 - \epsilon$ so that $\mathcal{I}_{1,2} \ll \mathbf{T}_b/3$, $\mathcal{I}_{j,b} \ll \mathbf{T}_b/3$, and $\Omega_j \ll \beta_j \mathbf{T}_b/3$, for $j = 1, 2$ and for every beacon $b \in \mathcal{B}$. Furthermore, ϵ can be chosen sufficiently small so that all expectations can be approximated by taking expectations over this set.

For each $b \in \mathcal{B}$, let $\text{SNR}_b = \mathbf{P}_b/\mathbf{P}_0$ be the effective output signal-to-noise ratio of the received signal from b . As shown in [57], the effective output SNR_b depends on the input SNR and the bandwidth-time product $\mathbf{W}_b \mathbf{T}_b$. Let

$$\lambda_b = \frac{8\pi^2 \mathbf{W}_b^2 \text{SNR}_b}{c^2}, \quad \text{and} \quad \epsilon_b = \frac{8\pi^2 \mathbf{T}_b^2 \text{SNR}_b}{c^2}. \quad (18)$$

Furthermore, defining $\mathbf{u}_{j,b} = (\mathbf{p}_j - \mathbf{p}_b)/(\|\mathbf{p}_j - \mathbf{p}_b\|)$ and $\mathbf{w}_{j,b} = f_b(\mathbf{I} - \mathbf{u}_{j,b} \mathbf{u}_{j,b}^T) \mathbf{v}_j / (\|\mathbf{p}_j - \mathbf{p}_b\|)$, we have the following theorem, whose proof can be found in [53].

Theorem 1 *Suppose that Assumption 2 holds. Let $\boldsymbol{\phi}_b = [\mathbf{u}_{1,b}^T, -\mathbf{u}_{2,b}^T]^T$, $\boldsymbol{\phi}_s = [\mathbf{u}_{1,2}^T, -\mathbf{u}_{1,2}^T]^T$, $\boldsymbol{\rho}_b = [\mathbf{w}_{1,b}^T, -\mathbf{w}_{2,b}^T]^T$, and $\boldsymbol{\rho}_s = [\mathbf{w}_{1,2}^T, -\mathbf{w}_{1,2}^T]^T$. Treating the robot*

clock skews as nuisance parameters, the EFIM for estimating the robots' locations and velocities at robot S_1 , denoted as $\mathbf{F}_{e,s}$, is approximately given by

$$\mathbf{F}_{e,s} = \frac{1}{2} \sum_b \begin{bmatrix} \lambda_b(\Phi_b - \Xi) + \varepsilon_b(1 - \sigma_{\beta_b}^2)^{-1} \mathbf{P}_b & \varepsilon_b(1 - \sigma_{\beta_b}^2)^{-1} \Gamma_b \\ \varepsilon_b(1 - \sigma_{\beta_b}^2)^{-1} \Gamma_b^T & \varepsilon_b(1 - \sigma_{\beta_b}^2)^{-1} \Pi_b \end{bmatrix}, \quad (19)$$

where we let $\bar{\lambda}_b = \frac{\lambda_b}{\sum_{b'} \lambda_{b'}}$, $\sigma_s = 2[1 + (\sigma_{\beta_1}^2 - 3\sigma_{\beta_1}^2 \sigma_{\beta_2}^2)/(1 - 3\sigma_{\beta_2}^2 + \sigma_{\beta_1}^2 \sigma_{\beta_2}^2)]$, and

$$\begin{aligned} \Xi &= \sigma_s \sum_b \bar{\lambda}_b (\phi_b - \phi_s) \sum_b \bar{\lambda}_b (\phi_b - \phi_s)^T, \\ \Phi_b &= (\phi_b - \phi_s)(\phi_b - \phi_s)^T, \\ \mathbf{P}_b &= (\rho_b - \rho_s)(\rho_b - \rho_s)^T, \\ \Pi_b &= (f_b \phi_b - f_0 \phi_s)(f_b \phi_b - f_0 \phi_s)^T, \\ \Gamma_b &= (\rho_b - \rho_s)(f_b \phi_b - f_0 \phi_s)^T. \end{aligned}$$

Theorem 1 provides an approximate lower bound for the location and velocity estimation errors. However, since we have used the modified Bayesian CRLB, this bound is not tight. Nevertheless, the performance bound gives us various insights into the problem of location and velocity estimation, which we discuss below.

(1) We see from (19) that the EFIM consists of various ‘‘information matrix’’ components, which we describe in the following.

- The matrix Φ_b can be interpreted as the ranging direction matrix (RDM) associated with the directions $\mathbf{u}_{1,b} - \mathbf{u}_{1,2}$ and $\mathbf{u}_{2,b} - \mathbf{u}_{1,2}$. This is similar to the RDM introduced in [63], where the EFIM is derived in the case where a single robot localizes with the aid of synchronized anchors so that $\mathbf{u}_{1,2} = \mathbf{0}$. The RDM shows that each beacon provides only one-dimensional ranging information for each robot S_j , along the direction $\mathbf{u}_{j,b} - \mathbf{u}_{1,2}$ with a weight λ_b that can be interpreted as the *ranging information intensity*, which is a constant that depends only on beacon characteristics like SNR. We note that the beacon clock skews do not affect the λ_b or the RDM, which is intuitively correct as TDOA ranging is not affected by beacon clock skews.
- Let \mathbf{u}^\perp be the unit vector orthogonal to \mathbf{u} . The vectors ρ_b and ρ_s contain Doppler shift information in the directions $\mathbf{u}_{1,b}^\perp$ and $\mathbf{u}_{1,2}^\perp$ respectively. Therefore, the matrix \mathbf{P}_b can be interpreted as the relative Doppler information matrix associated with the directions $\mathbf{u}_{1,b}^\perp$, $\mathbf{u}_{2,b}^\perp$ and $\mathbf{u}_{1,2}^\perp$. This is intuitively appealing as all location information in the directions $\mathbf{u}_{1,b}$, $\mathbf{u}_{2,b}$, and $\mathbf{u}_{1,2}$ have already been captured in Φ_b so that \mathbf{P}_b contains additional information in directions orthogonal to these. Moreover, since Doppler shift is affected by beacon clock skews, we have a factor of $(1 - \sigma_{\beta_b}^2)^{-1}$ multiplied to \mathbf{P}_b in (19). We can also interpret ε_b as the *Doppler information intensity*.

- The term $f_b \phi_b$ is the rate of change of Doppler shift in the direction $\mathbf{u}_{j,b}$ w.r.t. \mathbf{v}_j . Therefore, the matrix $\mathbf{\Pi}_b$ contains information associated with how fast the Doppler shift along the directions $\mathbf{u}_{j,b}$ is changing w.r.t. that along $\mathbf{u}_{1,2}$. This is mainly useful for velocity estimation, and so do not appear in the CRLB derived in [63]. Beacon clock skews affect the information contained in $\mathbf{\Pi}_b$ and appears in the multiplicative factor $(1 - \sigma_{\beta_b}^2)^{-1}$ in (19).
- The term $\mathbf{\Xi}$ contains the weighted average ranging information from all beacons, with the weight of beacon b being λ_b normalized by the sum of all ranging information intensities. Since S_2 transmits all its received signals from the beacons to S_1 , we can interpret $\mathbf{\Xi}$ as the collective effect of robot clock asynchronism on the information transmitted from S_2 .

Notice that direction vectors, such as $\mathbf{u}_{j,b} - \mathbf{u}_{1,2}$ and its perpendicular counterpart, and the signal characteristics, including ranging information intensity λ_b and Doppler information intensity ϵ_b , play an equally important part in the estimation performance. As signals are transmitted by SOOP beacons, robots have no control over all signal characteristics. However, it can improve its localization performance by selecting a proper set of beacons and by moving with respect to its fellow robot such that a better geometry is formed. The performance bound in (19) hence provides a valuable guidance for beacon selection and robot movement in terms of localization geometry.

- (2) Consider a simpler scenario where two robots and all beacons are static, so that $\rho_b = \mathbf{0}$ and $\rho_s = \mathbf{0}$. The EFIM in (19) for robot locations reduces to

$$\mathbf{F}_{e,static} = \frac{1}{2} \sum_b \lambda_b (\Phi_b - \mathbf{\Xi}). \quad (20)$$

It is obvious that the designing of the vector $\phi_b - \phi_s$ holds the key to achieve the desired level of localization precision, which depends only on the relative position between robots and beacon. Two robots can hence cooperate with each other to create a proper topology for their localization.

- (3) From (19), it is clear that the EFIM for robot locations and velocities depend on neither the value of beacon clock offsets nor the value of robot clocks offsets. This suggests that there exists estimation algorithms that can eliminate *both* beacon and robot clock offsets. It can be shown that the TDOA procedure cancels out the beacon clock offsets.
- (4) Although the size of clock offsets do not affect the EFIM, we observe that there is loss of information whenever robot clocks are not synchronized. Consider an ideal case where both robots and beacons are static and synchronized, it can be shown that the EFIM is given by

$$\mathbf{F}_{e,syn} = \sum_b \lambda_b \left(\Phi_b + \begin{bmatrix} \mathbf{u}_{1,b} \mathbf{u}_{1,2}^T + \mathbf{u}_{1,2} \mathbf{u}_{1,b}^T & \mathbf{u}_{1,b} (\mathbf{u}_{2,b} - \mathbf{u}_{1,2})^T \\ (\mathbf{u}_{2,b} - \mathbf{u}_{1,2}) \mathbf{u}_{1,b}^T & \mathbf{0} \end{bmatrix} \right). \quad (21)$$

Therefore, the information loss due to robot clock offsets is given by a non-zero quantity $\mathbf{F}_{e,syn} - \mathbf{F}_{e,static}$, and it does not depend on the value of robot clock offsets, suggesting that clock offsets can be eliminated but at the price of a constant information loss. This is consistent with our proposed algorithm, where the location estimation in Sect. 3.2.1 is not affected by any clock offsets but more measurements are required to calculate the DTDOA.

- (5) The EFIM however depends on clocks skews of beacons and robots. The effect of the robot clock skew is summarised by the term σ_s , and it appears as a collective effect from both robots due to the fact that two robots need to exchange their perspective received signals. The beacon clock skew on the other hand has a direct impact on the frequency of the transmitted signal, which leads to a frequency bias eventually in the FDOA measurements. In Sect. 4, we conduct simulations to further verify the effect of the clock skews, and it shows that they have limited impact.

Theorem 1 provides an analytical expression for assessing the estimation accuracy of a team of two robots working with cooperative scheme in Sect. 2. When the team has more than two robots, each robot may have more than one neighbouring robot to cooperate with, and its localization accuracy will further depend on the relative positions among the set of robots. Such relationship was defined as relative position measurement graph (RPMG) in [33], and it can be considered as an intra-network factor while the relation between robots and beacons in Theorem 1 serves as an inter-network factor. The combined effect of two factors is obviously complicated, and the result in Theorem 1 provides a building block for the analysis.

4 Simulation Results

Simulations are carried out to verify the CRLB. Two robots S_1 and S_2 are 1 km apart, with both robots moving in reverse direction, each at a speed of 100 km/h. N beacons are randomly scattered in a 10 Km by 10 km area under the assumption that robots are in the convex hull of the beacons. One example is to place all beacons in a circle and start the movement of two robots from the center. The clock skews β_m are drawn from Gamma distributions with $\mathbb{E}\{\beta_m\} = 1$ and varying standard deviations σ_{β_m} for $m \in \mathcal{S} \cup \mathcal{B}$. The signal bandwidth is 200 kHz and the integration time is 1 s. The measurement noise for TDOA and FDOA can then be calculated using standard expressions in [57]. For each set of parameters, 10,000 simulation runs are performed, a geometry of beacons is drawn randomly in each run, and the initial guess in each run is drawn randomly from the region bounded by the beacon positions.

Figure 3 shows the modified Bayesian CRLB under various number of beacons and measurement noise, where the standard deviations of clock skews are set to 10^{-6} . The DTDOA algorithm in [53] is also shown for comparison. Since approximations have been made in the algorithm and we used a looser modified CRLB, there exists

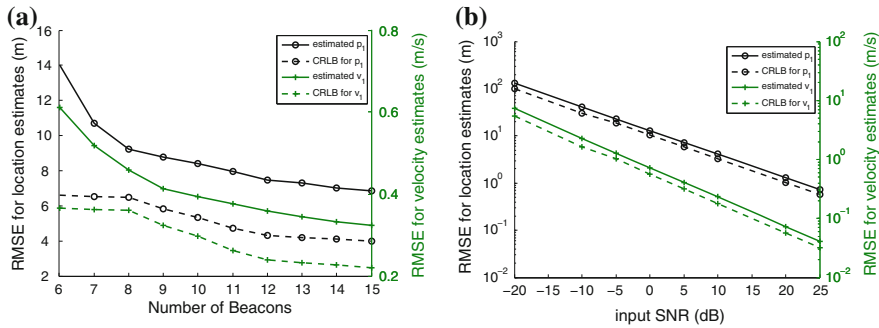


Fig. 3 Effect of measurement noise on CRLB and the DTODA algorithm in [53]. **a** Number of Beacons. **b** Input SNR (dB)

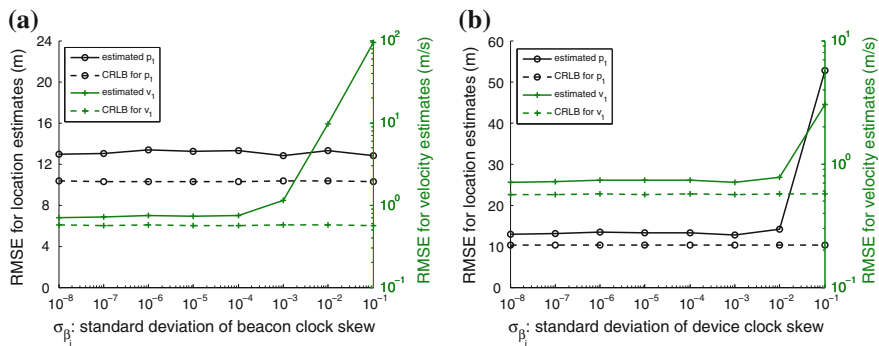


Fig. 4 Effect of clock asynchronism on CRLB and the DTODA algorithm in [53]. **a** σ_{β_j} : Standard deviation of beacon clock skew. **b** σ_{β_j} : Standard deviation of device clock skew

a performance gap between our algorithm and the bound. When the number of beacons is larger than 8, the RMSE gap for the location and velocity estimates is almost constant at about 3.08 m and 0.17 m/s, respectively.

Figure 4 investigates the effect of clock asynchronism on the location and velocity estimation, where the number of beacons is set to be 8 and the input SNR is 0 dB. As shown in Fig. 4a, b, the modified Bayesian CRLB is almost constant over a wide range of robot and beacon clock skews, showing that clock skews have limited impact in most practical applications. The performance of the algorithm deteriorates for larger clock skews, especially for velocity estimation with varying beacon clock skews. This is because we have approximated $\delta_b^{\xi} \approx 0$ in (12), which holds only for small δ_{β_b} and δ_{β_j} . As the clock skews increase, this approximation is violated and the estimation error increases. However, we note that when the beacon clock skew is smaller than 10^{-4} and the robot clock skew is smaller than 10^{-2} , our proposed algorithm achieves estimation accuracy close to the modified Bayesian CRLB. As the typical value for standard deviations of crystal oscillators ranges from 10^{-8} to 10^{-4} , it is expected that our algorithm is robust in most situations.

5 Conclusions

In this chapter, we have investigated the self-localization problem using SOOP for a robotic platform with autonomous robots. We reviewed existing cooperative localization schemes for the robotic platform, and provided a survey on the state-of-the-art localization techniques using SOOP. By assuming that robots are loosely synchronized and each robot can communicate with its neighbouring robots by exchanging a short segment of its received signal, we proposed a cooperative localization scheme for the robots. Although the exchange of signals increases the communication burden for each robot, the proposed scheme enables the robots to use SOOP for localization, makes full use of existing infrastructure and allows exploitation of a wide variety of signals in different frequency bands. As the cooperation between two robots forms the building block for the proposed scheme, we analyzed its performance limit by deriving the EFIM of the modified Bayesian CRLB. The EFIM reveals the effect of various factors, such as relative direction and clock asynchronism, on the localization accuracy.

References

1. Cox, I.J.: Blanche: Position estimation for an autonomous robot vehicle. In: IEEE/RSJ International Workshop on Intelligent Robots and Systems, Sept 1989, pp. 432–439
2. Win, M.Z., Conti, A., Mazuelas, S., Shen, Y., Gifford, W.M., Dardari, D., Chiani, M.: Network localization and navigation via cooperation. *IEEE Commun. Mag.* **49**(5), 56–62 (2011)
3. Borenstein, J., Everett, H.R., Feng, L., Wehe, D.: Mobile robot positioning—sensors and techniques. *J. Rob. Syst.* **14**(4), 231–249 (1997)
4. Borenstein, J., Feng, L., Borenstein, C.J.: Measurement and correction of systematic odometry errors in mobile robots. *IEEE Trans. Rob. Autom.* **12**(6), 869–880 (1996)
5. Leonard, J., Durrant-Whyte, H.: Mobile robot localization by tracking geometric beacons. *IEEE Trans. Rob. Autom.* **7**(3), 376–382 (1991), Jun
6. Basic Guide to Advanced Navigation, NATO Science and Technology Organization, Feb 2010
7. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), Aug 2000, Boston
8. Oman, J.: Crafting a cricket batsman robot, Master Thesis, Lulea University of Technology, 2005
9. Priyantha, N.B.: The cricket indoor location system, Master's Thesis, Massachusetts Institute of Technology, 2005
10. Harter, A., Hopper, A., Steggle, P., Ward, A., Webster, P.: The anatomy of a context-aware application. *Wirel. Netw.* **8**(2–3), 187–197 (2002)
11. Fukuju, Y., Minami, M., Morikawa, H., Aoyama, T.: DOLPHIN: an autonomous indoor positioning system in ubiquitous computing environment. In: IEEE Workshop on Software Technologies for Future Embedded Systems, May 2003, pp. 53–56
12. Minami, M., Fukuju, Y., Hirasawa, K., Yokoyama, S., Mizumachi, M., Morikawa, H., Aoyama, T.: DOLPHIN: a practical approach for implementing a fully distributed indoor ultrasonic positioning systems. In: Davies, N., Mynatt, E., Siio, I. (eds.) *UbiComp 2004: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, Vol. 3205, pp. 347–365. Springer, Berlin, Heidelberg (2004)

13. Bahl, P., Padmanabhan, V.N.: RADAR: an in-building RF-based user location and tracking systems. In: IEEE International Conference on Computer Communications (INFOCOMM'00), 2000, pp. 775–784
14. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. *ACM Trans. Inf. Syst.* **10**(1), 91–102 (1992)
15. Ubisense Real-Time Location Fact Sheet. <http://www.ubisense.net>. Accessed 2 Mar 2010
16. Merry, L.A., Faragher, R.M., Scheduling, S.: Comparison of opportunistic signals for localisation. In: 7th IFAC Symposium on Intelligent Autonomous Vehicles (2010), vol. 7, no. 1, Lecce, Italy
17. Parker, L.: English Current state of the art in distributed autonomous mobile robotics. In: Parker, L., Bekey, G., Barhen, J. (eds.) *English Distributed Autonomous Robotic Systems*, Vol. 4, pp. 3–12. Springer, Japan (2000)
18. Kurazume, R., Nagata, S., Hirose, S.: Cooperative positioning with multiple robots. In: IEEE International Conference on Robotics and Automation, 1994, vol. 2, pp. 1250–1257
19. Kurazume, R., Hirose, S.: English an experimental study of a cooperative positioning system. *Engl. Auton. Robots* **8**(1), 43–52 (2000)
20. Grabowski, R., Navarro-Serment, L., Paredis, C., Khosla, P.: Heterogeneous teams of modular robots for mapping and exploration. *Engl. Auton. Robots* **8**(3), 293–308 (2000)
21. Rekleitis, I., Dudek, G., Miliotis, E.: English Multi-robot collaboration for robust exploration. *Engl. Ann. Math. Artif. Intell.* **31**(1–4), 7–40 (2001)
22. Rekleitis, I., Dudek, G., Miliotis, E.: Probabilistic cooperative localization and mapping in practice. In: IEEE International Conference on Robotics and Automation (ICRA'03), Sept 2003, vol. 2, pp. 1907–1912
23. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. *Engl. Auton. Robots* **8**(3), 325–344 (2000)
24. Caglioti, V., Citterio, A., Fossati, A.: Cooperative, distributed localization in multi-robot systems: a minimum-entropy approach. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), June 2006, pp. 25–30
25. Roumeliotis, S., Bekey, G.A.: Collective localization: a distributed kalman filter approach to localization of groups of mobile robots. In: IEEE International Conference on Robotics and Automation (ICRA'00), 2000, vol. 3, pp. 2958–2965
26. Roumeliotis, S., Bekey, G.A.: Distributed multirobot localization. *IEEE Trans. Robot. Autom.* **18**(5), 781–795 (2002)
27. Martinelli, A.: Improving the precision on multi robot localization by using a series of filters hierarchically distributed. In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 1053–1058
28. Howard, A., Mataric, M., Sukhatme, G.: Localization for mobile robot teams using maximum likelihood estimation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*, vol. 1, pp. 434–439
29. Howard, A., Mataric, M., Sukhatme, G.: Localization for mobile robot teams: a distributed MLE approach. In: Siciliano, B., Dario, P. (eds.) *English Experimental Robotics VIII*, ser. Springer Tracts in Advanced Robotics, Springer, Berlin, Heidelberg, 2003, vol. 5, pp. 146–155
30. Nerurkar, E.D., Roumeliotis, S., Martinelli, A.: Distributed maximum a posteriori estimation for multi-robot cooperative localization. In: *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009, pp. 1402–1409
31. Trawny, N., Roumeliotis, S., Giannakis, G.: Cooperative multi-robot localization under communication constraints. In: *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009, pp. 4394–4400
32. Roumeliotis, S., Rekleitis, I.: Propagation of uncertainty in cooperative multirobot localization: analysis and experimental results. *Engl. Auton. Robots* **17**(1), 41–54 (2004)
33. Mourikis, A., Roumeliotis, S.: Performance analysis of multirobot cooperative localization. *IEEE Trans. Robot.* **22**(4), 666–681 (2006)
34. Howard, A., Siddiqi, S., Sukhatme, G.S.: An experimental study of localization using wireless ethernet. In: *4th International Conference on Field and Service Robotics*, 2003

35. Ladd, A.M., Bekris, K.E., Rudys, A., Kavradi, L.E., Wallach, D.S.: Robotics-based location sensing using wireless ethernet. *Wirel. Netw.* **11**, 189–204 (2005)
36. Huang, J., Millman, D., Quigley, M., Stavens, D., Thrun, S., Aggarwal, A.: Efficient, generalized indoor WiFi graphSLAM. In: *IEEE International Conference on Robotics and Automation (ICRA'11)*, May 2011, pp. 1038–1043
37. Li, B., Gallagher, T., Dempster, A., Rizos, C.: How feasible is the use of magnetic field alone for indoor positioning? In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, Nov 2012, pp. 1–9
38. Jimenez, A., Zampella, F., Seco, F.: Light-matching: a new signal of opportunity for pedestrian indoor navigation. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN'13)*, Oct 2013, pp. 1–10
39. Schwaighofer, A., Grigoras, M., Tresp, V., Hoffmann, C.: *GPPS: a Gaussian process positioning system for cellular networks*. In: *IN NIPS*. MIT Press, 2004
40. Del Peral-Rosado, J., Lopez-Salcedo, J., Seco-Granados, G., Zanier, F., Crisci, M.: Achievable localization accuracy of the positioning reference signal of 3GPP LTE. In: *International Conference on Localization and GNSS (ICL-GNSS'12)*, June 2012, pp. 1–6
41. Dammann, A., Sand, S., Raulefs, R.: Signals of opportunity in mobile radio positioning. In: *Proceedings of the 20th European Signal Processing Conference*, Aug 2012, pp. 549–553
42. Moghtadaee, V., Dempster, A., Lim, S.: Indoor localization using FM radio signals: a fingerprinting approach. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, Sept 2011, pp. 1–7
43. Dong, Q., Dargie, W.: Evaluation of the reliability of RSSI for indoor localization. In: *International Conference on Wireless Communications in Unusual and Confined Areas (ICW-CUCA'12)*, Aug 2012, pp. 1–6
44. LaMarca, A., Hightower, J., Smith, I., Consolvo, S.: Self-mapping in 802.11 location systems. In: *Beigl, M., Intille, S., Rekimoto, J., Tokuda, H. (eds.) UbiComp 2005: Ubiquitous Computing*, ser. *Lecture Notes in Computer Science*, 2005, vol. 3660, pp. 87–104. Springer, Berlin, Heidelberg
45. Haeberlen, A., Flannery, E., Ladd, A.M., Rudys, A., Wallach, D.S., Kavradi, L.E.: Practical robust localization over large-scale 802.11 wireless networks. In: *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom'04)*, 2004, pp. 70–84. ACM, New York
46. Ferris, B., Haehnel, D., Fox, D.: Gaussian processes for signal strength-based location estimation. In: *Proceedings of Robotics: Science and Systems*, Philadelphia, Aug 2006
47. Ferris, B., Fox, D., Lawrence, N.: WiFi-SLAM using Gaussian process latent variable models. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, San Francisco, 2007, pp. 2480–2485
48. Lim, H., Kung, L.-C., Hou, J.C., Luo, H.: Zero-configuration indoor localization over IEEE 802.11 wireless infrastructure. *Wirel. Netw.* **16**(2), 405–420 (2010)
49. Gallagher, T., Li, B., Dempster, A., Rizos, C.: A sector-based campus-wide indoor positioning system. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, Sept 2010, pp. 1–8
50. Li, B., Wang, Y., Lee, H., Dempster, A., Rizos, C.: Method for yielding a database of location fingerprints in WLAN. *IEE Proc. Commun.* **152**(5), 580–586 (2005). Oct
51. Leng, M., Wu, Y.-C.: Distributed clock synchronization for wireless sensor networks using belief propagation. *IEEE Trans. Signal Process.* **59**(11), 5404–5414 (2011)
52. Leng, M., Tay, W.P., Quek, T.Q.S.: Cooperative and distributed localization for wireless sensor networks in multipath environments. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 3125–3128
53. Leng, M., Tay, W.P., See, C., Razul, S., Win, M.: Modified CRLB for cooperative geolocation of two devices using signals of opportunity. *IEEE Trans. Wirel. Commun.* **13**(7), 3636–3649 (2014)
54. Sun, M., Ho, K.C.: An asymptotically efficient estimator for TDOA and FDOA positioning of multiple disjoint sources in the presence of sensor location uncertainties. *IEEE Trans. Signal Process.* **59**(7), 3434–3440 (2011)

55. Musicki, D., Kaune, R., Koch, W.: Mobile emitter geolocation and tracking using TDOA and FDOA measurements. *IEEE Trans. Signal Process.* **58**(3), 1863–1874 (2010)
56. Wei, H.-W., Peng, R., Wan, Q., Chen, Z.-X., Ye, S.-F.: Multidimensional scaling analysis for passive moving target localization with TDOA and FDOA measurements. *IEEE Trans. Signal Process.* **58**(3), 1677–1688 (2010)
57. Stein, S.: Algorithms for ambiguity function processing. *IEEE Trans. Acoust., Speech, Signal Process.* **29**(3), 588–599 (1981)
58. Kay, S.M.: *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, Englewood Cliffs (1993)
59. Harry, K.L.B., Van Trees, L. (ed.): *Bayesian Bounds for Parameter Estimation and Nonlinear Filtering/Tracking*. Wiley-IEEE Press, New York (2007)
60. Moeneclaey, M.: On the true and the modified Cramer-Rao bounds for the estimation of a scalar parameter in the presence of nuisance parameters. *IEEE Trans. Commun.* **46**(11), 1536–1544 (1998)
61. Gini, F., Reggiannini, R., Mengali, U.: The modified Cramer-Rao bound in vector parameter estimation. *IEEE Trans. Commun.* **46**(1), 52–60 (1998)
62. Hlawatsch, F., Auger, F. (eds.) *Time-Frequency Analysis: Concepts and Methods*. Wiley, London (2008)
63. Shen, Y., Win, M.: Fundamental limits of wideband localization—part I: a general framework. *IEEE Trans. Inf. Theory* **56**(10), 4956–4980 (2010)

Part III
Security and Dependability

Security in Mobile Wireless Sensor Networks: Attacks and Defenses

Amrita Ghosal and Subir Halder

Abstract In recent years wireless sensor networks (WSNs) are fast emerging as an important domain for both commercial and personal use. The advancement in robotics has enabled exploring another domain in WSNs i.e., mobile WSNs (MWSNs). A MWSN consists of a collection of nodes that can move on their own and interact with the physical environment. Several applications demand the need for mobility in nodes which, in general are static. Due to the nature of deployment of the nodes coupled with their resource constraints, providing security to such MWSNs have gained a prime importance. Also, they can be deployed in physically inaccessible environment as well as critical areas, and therefore the need to make them secure is very important. Mobility of nodes in MWSNs makes them more vulnerable to attacks by adversaries. Many works have been conducted in recent past where various promising solutions have been provided for detecting the attack, diagnosing the adversary nodes, and nullifying their capabilities for further damage in MWSNs. To start with, this chapter presents the need for MWSNs followed by security objectives, key issues and inherent challenges faced by these networks. Existing works dealing with basic security features and the different attacks faced by MWSNs are discussed. Finally, we give an insight into the possible directions for future work in securing MWSNs.

1 Introduction

Advances in device technology, radio transceiver designs and integrated circuits along with efficient network protocols have enabled the emergence of WSNs. Now a days WSNs have been widely considered as one of the most important technologies

A. Ghosal (✉) · S. Halder
Department of Computer Science and Engineering,
Dr. B. C. Roy Engineering College, Durgapur, India
e-mail: ghosal_amrita@yahoo.com

S. Halder
e-mail: subir_ece@rediffmail.com

for the twenty first century. A WSN consists of hundreds or thousands of resource constraint (with respect to memory, battery power, storage and processing capability) sensor nodes (we use the terms ‘sensor node’, ‘sensor’ and ‘node’ interchangeably in this chapter). These sensor nodes perform three basic tasks: (i) sample a physical quantity from the surrounding environment, (ii) process (and possibly store) the acquired data, and (iii) transfer them through wireless communication to a data collection point called base station or sink.

The traditional WSN architectures are based on the assumption that the network is dense, so that any two nodes can communicate with each other through multi-hop paths. As a consequence, in most cases, the sensor nodes are assumed to be static, and mobility is not considered as an option. Several classes of WSNs were explored in the recent past and one such class is MWSN. A MWSN consists of sensor nodes which have the ability of being mobile [1]. Mobile nodes have the ability to sense, compute, and communicate like static nodes. Mobility is induced by using mobilizers with sensors for altering their position [2] or by use of springs [3, 4] or wheels [5] that help the sensor nodes to self propel. Also, they can be attached with transporters like vehicles, robots etc. [6] or movement may also occur due to the environment [7]. MWSN is fast emerging as an important research paradigm mainly due to their applicability in real life settings. Initially, mobility was considered to be highly difficult with respect to implementation in WSNs [8] but recent studies have demonstrated that mobility can alleviate several problems in static WSNs such as coverage, connectivity and energy consumption [7]. Mobility also allows nodes for targeting and tracking events involving movement such as vehicles, chemical clouds etc. [9]. A few of the applications where mobile sensor nodes can be used are for environmental monitoring, agriculture production monitoring, smart city, tracking moving objects, battlefield monitoring etc. [10–12].

MWSN came into the picture due to several problems that were faced by static WSNs. Some of these problems are listed below:

- Complete coverage and connectivity is not ensured during initial deployment of the sensor nodes leading to partitioning of the whole area into small unconnected networks. Also, dynamic change of scenarios due to environment related factors and obstacles can also aggravate the problem further.
- Nodes in WSN are highly energy constrained as they are mostly battery powered and very much susceptible to errors. Generally, the batteries in such nodes are irreplaceable. Untimely death of nodes due to battery drainage may create energy holes that disturb the network’s operation.
- In certain scenarios WSNs may be needed to fulfill multiple objective missions under various conditions [13]. For example in an object tracking application, adequate number of nodes are deployed along the track of the target whereas in a boundary detection mission sufficient nodes should be placed along the pre-described perimeter. Satisfying all these requirements by deploying several number of nodes is really very difficult as providing for all possible combinations of mission requirements is not economically feasible. There are some applications that may require implementation of expensive and sophisticated nodes. For

example, in a military application, nodes having imaging and pressure sensors are needed to be deployed for tracking enemies along the border. This requirement is very much costly as all the nodes should have the ability to capture images and sense pressure making it very much infeasible for static WSNs.

All these factors make mobility a viable option for networks such as WSNs as introducing mobility will increase the ability of the network to support multiple missions and also prevent the occurrence of the problems mentioned above.

1.1 Motivation and Contributions of the Book Chapter

More recently, advances in robotics have made it possible to develop a variety of new architectures for autonomous wireless networks of sensor nodes i.e., MWSNs. A compendium of knowledge representing rich collection on security issues in WSNs can be found in the recent past literatures [14–17]. In all these works, researchers have considered that nodes including sink and/or malicious nodes are static i.e., nodes cannot move. Further, many recent works [18–20] have been published where the security of mobile devices, mobile agents based wireless networks are extensively studied. Nevertheless, these techniques are not suitable for MWSN due to their unique characteristics. Contrary to mobile devices, mobile agents based wireless networks, which have more energy resources, MWSN have severely constrained resources. In MWSN, more importantly, due to mobility the trust relationships among the mobile nodes changes quickly which enforce great challenge in the security of the mobile nodes in MWSNs. Further, as the nodes are mobile in MWSNs, therefore, they usually exchange more messages among themselves, resulting in exposing themselves to attackers that may attack the mobile nodes in motion. Furthermore, node mobility can make implementation of detection mechanisms worse since the correct behaviour of the nodes is location and neighbourhood dependent. Particular scenarios where mobile adversaries are present, it is very much necessary for the nodes to be mobile to prevent degradation in network performance. Under these constraints, deploying such new technology (i.e., MWSN) without security in mind would be unreasonably dangerous compared to the mobile devices, mobile agents based wireless networks. All these factors motivate us to take up this work. In this chapter, we address the needs of MWSN and discuss the key issues and inherent challenges faced by MWSN with respect to security. We also attempt to review the existing state-of-the-art defense and detection mechanisms for attacks prevalent in MWSNs. Finally, we give an insight into the possible directions for future work in securing MWSNs.

1.2 Chapter Organization

This chapter has been organized as follows. In Sect. 2, we briefly explain the technical preliminaries, security objective, key issues and main challenges related to

the designing and implementation of a security scheme in MWSN. In Sect. 3, existing state-of-the-art defense and detection mechanisms are studied. Section 4 draws possible open research issues. Finally, Sect. 5 concludes this chapter.

2 Security Objective, Issues and Challenges in MWSNs

In this section, we first introduce the technical preliminaries including categories and advantages of MWSNs. We mention security objectives and key issues of designing security schemes, which are very essential aspects in any security system including mobile nodes and/or sink in MWSNs. Finally, the inherent challenges incurred by the MWSNs are also discussed.

2.1 Background

A MWSN consists of a collection of sensor nodes, essentially small robots that can move on their own and interact with the physical environment. Mobile nodes have the ability to sense, compute, and communicate like static nodes. A key difference with static nodes is that mobile nodes have the ability to reposition and reorganize themselves in the network. In the existing literature, MWSNs are mainly categorized into two types, based on type of communication and role of nodes.

Type of Communication: Based on the type of communication, MWSNs are classified into two kinds [7] viz. infrastructure network and infrastructureless network. In infrastructure network, the mobile unit remains connected with the nearest sink that lies within its communication radius. On the contrary, in infrastructureless network, all nodes are capable of moving. Such networks are self organizing in nature, having the capability of establishing communication between them.

Role of Nodes: MWSNs at the node level can be divided into four categories viz. mobile embedded sensor, mobile actuated sensor, data mule and access point. In mobile embedded sensor, the movement of the mobile embedded nodes is not controlled by themselves; instead their motion is guided by some external force, for example, when attached with an animal [21]. While in mobile actuated sensor, sensor nodes have the capability of locomotion [5, 22] that allows them to move freely around the sensing region. Therefore, the deployment will be more specific, ensuring maximum coverage and better performance with respect to targeting. On the contrary, in some cases a mobile device referred to as data mules [23] collect data from nodes and transmit them to the sink. So nodes need not be mobile. Finally, in access point, mobile nodes can behave as access points in case of sparse networks or when the network connectivity [24] is broken due to nodes dropping off from the network.

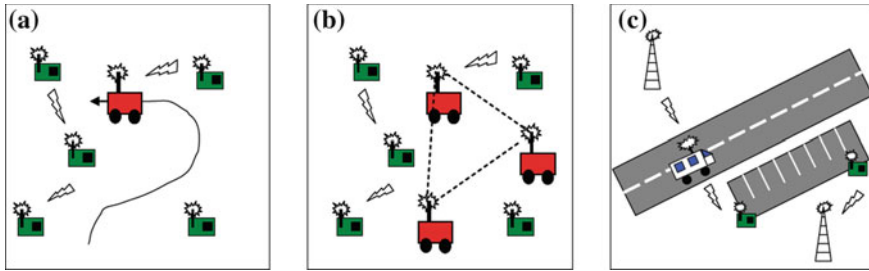


Fig. 1 MWSNs Architectures [7]. a Flat. b Two-tier. c Three-tier

2.1.1 Architecture

Architectures for MWSNs can be classified as flat, two-tier and three-tier architectures [7]. Each of these categories is briefly described below:

Flat Architecture: This type of architecture is also referred to as planar architecture (Fig. 1a). This network architecture consists of a set of heterogeneous devices where communication is performed in an ad-hoc way [7]. Examples of systems that utilize flat or planar architecture are basic navigation systems [25] etc.

Two-tier Architecture: The two-tier architecture (Fig. 1b) consists of two kinds of nodes e.g., a set of stationary nodes and a set of mobile nodes [7]. The mobile nodes are responsible for forming either an overlay network or act as data mules so as to assist data movement throughout the network. The overlay network may consist of mobile devices that possess more processing ability, larger communication range and higher bandwidth. The network density of the nodes in the overlay network is arranged in such a manner that the nodes always remain connected and there is no possibility of any breakage of connection in the network. In the worst case even if any network disconnection occurs, the mobile nodes relocate themselves so as to re-establish connectivity and ensure that packets reach their final destination. One example of a system that uses two-tier approach is NavMote system [26].

Three-tier Architecture: In this architecture a group of stationary nodes transfer data to a set of mobile devices, which further transmit the same to access points [7]. The three-tier architecture (Fig. 1c) is designed to cover wider areas and provide compatibility to a variety of applications.

2.1.2 Advantage

Research in recent years have revealed that MWSNs perform better with respect to WSNs [2, 4, 7, 27] as they provide certain advantages which are given below:

Architecture: The architecture for a MWSN can be sparse in nature instead of dense as needed for static WSNs [1]. This reduces the overhead of the amount of nodes required for design of the network.

Network Reconstruction: In case of MWSNs if nodes in the network are disconnected due to node failure or energy depletion, the nodes can once again reconstruct the network. But in case of static WSN if node failure or energy depletion in nodes occur, the network is disconnected and there is no possibility of once again self reorganizing the network.

Enhancement of Network Lifetime: Mobility in sensor nodes enables transmission to occur in a scattered manner and energy dissipation more efficient, thereby eliminating the problem of occurrence of energy holes near the sink. But in static WSN as movement is not possible or restricted there is high probability of formation of energy holes leading to decrease in network lifetime. Also use of mobile sink in MWSNs solves the problem of energy holes and leads to increase in network lifetime in such networks [28, 29].

Increase in Channel Capacity: It has been found through experiments that MWSNs have capacity gains 3–5 times more than that of static WSNs [27] when the number of mobile sinks increases linearly with the increase in number of sensor nodes. Mobility also enhances channel capacity as more communication paths are created and the number of hops to be covered by the data/message to reach the destination is reduced [30].

Enhancement in Coverage and Targeting: As node deployment in most cases is random, so they may be required to move to a place providing better coverage or more proximity with respect to targeting. These criteria vary with the different application needs in WSN. It is evident that redeploying or rearranging sensor nodes in remote areas is very difficult but when sensor nodes are mobile, redeployment becomes easier. Adding mobile elements into WSNs improves coverage [24] and thereby the utility of sensor network deployment.

2.2 Security Objective

Security is one of the crucial aspects in any real-time application and in respect to MWSNs, it is not altogether different from those of other networks. It can greatly affect the network performance, especially in the data exchange phase. Therefore, considering the security attributes while designing a protocol for MWSN is a must. Due to the unique characteristics of MWSNs mentioned above, these networks are open to different types of attacks. Hence, before deploying a MWSN, security issues for the particular network need to be clarified. In dealing with network security, we will explain the following security services or requirements that effective security architectures must ensure:

- **Authentication** ensures that communication from one node to another is genuine. In other words, it ensures that a malicious node cannot masquerade as a trusted network node.

- **Confidentiality** ensures that a given message cannot be understood by anyone other than its (their) desired recipient(s). Confidentiality is typically enabled by applying symmetric or asymmetric encryption.
- **Integrity** ensures that the contents of a message sent from one node to another node are not modified by any malicious node during its transmission.
- **Availability** ensures that the desired network services are available whenever they are expected, inspite of presence of attacks.
- **Non-repudiation** is the ability to ensure that a node cannot deny sending a message that it originated. Digital signatures may be used to ensure this.

2.3 Key Issues

The particular requirements of security schemes for MWSNs generally depend on the nature of applications, constraints imposed by hardware and network infrastructure. Based on these, some of the specific issues concerning the design of the security scheme are as follows:

Amorphous: Node mobility including the sink and wireless connectivity allow nodes to enter and leave the network instinctively, to form and break communication links unintentionally [31]. Therefore, the network topology has no fixed form regarding both its size and shape, i.e. it changes frequently. Therefore, an essential issue while designing any robust security scheme for MWSNs is to take communication link failure into account.

Distribution: Security schemes may be totally distributed, partially distributed and centralized [32]. Each of this distribution of security scheme has strengths and limitations. Hence, it is utmost important while designing a security scheme for MWSNs to choose the appropriate distribution for making the scheme efficient and robust.

Communication and Computation Requirements: Communication between a mobile or static node and a mobile sink can provide significant benefits such as time synchronization [31]. However, a fundamental issue in MWSNs is the minimization of communication requirements in the mobile nodes to conserve energy. Further, mobile nodes have limited storage and weak computational capability. Therefore, highly complex security solutions, such as symmetric or asymmetric data encryption, are difficult to implement. This introduces unique considerations for designing the security scheme as well.

Topology: As the nodes are mobile, it is desirable that authentication and key exchanges do not depend on additional messages. Also, all the required key materials and cryptographic functions must be present on the nodes. Faster execution of cryptographic algorithms is desired to maintain the real time property of such networks. The security architecture should be designed keeping in mind the scalability factor of the network.

2.4 Inherent Challenges

In MWSNs, the movement of nodes including the sink is generally influenced by some factors or for the demand of a particular application [1]. To be more specific, nodes change their behaviour according to the actual role in the network. In a broader sense, threats to MWSNs can be either application dependent or application independent. Therefore, the inherent challenges for designing an efficient security scheme of MWSNs are as follows:

Trajectory Planning: In MWSNs, node mobility can make implementation of detection mechanisms worse since the correct behaviour of the nodes is location dependent [1, 29]. Mobile nodes trajectory has to be properly planned so as to be shortest in length as well as should be quick to provide accurate detection of adversary. Due to random node dropping, node placement pattern is not known a priori. In a dynamic environment, even if the initial pattern was known, the final node distribution may be different (e.g., moved by wind or animals). Trajectory of mobile node including sink thus, is the key challenge for the security scheme of MWSN and it should be planned on the fly rather than beforehand.

Infrastructureless Environment: MWSNs are free from any specialized infrastructure such as central servers, and fixed routers [7]. Above all, nodes are generally deployed in some inaccessible terrain or areas where infrastructures are very less. In order to detect adversaries, the security solutions should rely on a distributed cooperative scheme instead of a centralized one.

Dynamic Topology: In MWSN, nodes arbitrarily change their positions resulting in a highly dynamic topology causing wireless links or routes to be broken and re-establish on-the-fly [1]. Therefore, one of the major challenge for devising any security scheme is the dynamic topology characteristic of MWSN.

Resource Constraints: To enable cooperation among mobile nodes for detecting adversaries, information exchange between neighbouring mobile nodes adds to energy consumption and bandwidth occupancy [26]. In addition, as the nodes are mobile in MWSN, additional power for mobility is needed. Therefore, in MWSNs one of the biggest challenges while designing security solution is energy conservation.

Table 1 presents a summary of the main challenges faced by the security schemes in MWSNs and their possible resolution. It is worth noting that there is no single solution that combats all the challenges. The choice of the solution depends on the application, scenario, and available resources.

3 Security in MWSNs

MWSNs are much more vulnerable to security breaches and attacks by adversaries than static WSNs. It is mainly due to the fact that nodes are usually deployed in uncontrolled and untrusted locations that are prone to attacks. Further, more mobil-

Table 1 Challenges and their probable solution in designing security solutions

Challenge	Highlight of probable solution
Trajectory planning	Must be planned on the fly rather than beforehand to optimize the traversing path
Infrastructureless environment	Distributed cooperative security scheme is preferred more than centralized scheme
Dynamic topology	Additional efforts are required to alleviate the impact of wireless link failures and re-establishment
Resource constraint	Additional efforts are required to reduce communication cost, computation cost

ity means more exchange of messages exposing them to both passive and active attacks whereas, static WSNs are less prone to passive and active attacks, mainly to passive attacks such as traffic analysis attack. Some works in MWSNs have considered mechanisms for defense, detection or mitigation of particular attack(s) that are prevalent in MWSNs while others have considered providing basic security features such as authentication, integrity, confidentiality. The section given below illustrates the works that deal with mechanisms to handle attacks in MWSNs. In another section, works that ensure providing basic security features to MWSNs.

3.1 Security Attacks and Countermeasures

In general WSNs are very much prone to attacks mainly due to their broadcast nature of communication coupled with their deployment in unattended and hostile environment. Likewise, attacks in MWSNs are also very much widespread due to the reasons mentioned above along with their mobility feature [17]. In this section, we discuss the works that have considered defense and detection of attacks that are launched by adversaries in MWSNs. MWSN is an emergent topic and to the best of our knowledge the security aspect of it has not been dealt with much. So, we could find to our best possible effort only a handful of works in MWSNs that have considered basic security or attacks (describe in Sect. 3.2). Mainly, we could find works on the following attacks in MWSNs- node capture attack and node clone/replication attack. Defense mechanisms for other attacks such as wormhole attack, denial of service (DoS) attack, message corruption attack etc. are proposed in one or two works. We have tried to accommodate the works dealing with attacks in MWSNs to our best possible extent in the discussion that follows.

3.1.1 Node Capture Attack

In node capture attack, an adversary captures one or more nodes and utilizes the captured node(s) to launch further attacks. This section describes the works that deal with the detection followed by defense of node capture attack in MWSNs.

Detection Mechanism: Existing node capture attack detection mechanisms proposed for MWSNs are described below.

Conti et al. [33] proposed a mechanism for efficient detection of node capture attack in MWSNs. Two solutions were proposed by the authors- Simple Distributed Detection (SDD) and Cooperative Distributed Detection (CDD). SDD does not necessitate very clear information exchange among nodes for local detection of node capture attack whereas CDD is a more sophisticated approach that considers use of local node cooperation along with mobility for proper detection of this attack. A specific time period is considered within which if a node does not re-meet another node then it is considered that probably the other node has been captured. In case of SDD, each node maintains a tracking system where it stores the meeting time with other nodes in the network within its communication range. If at any point of time the threshold (i.e., the time set within which the nodes should re-meet) is crossed and if any node does not receive any communication with any previously interacted node, an alarm is generated by the node. If a certain number of alarms are generated for a particular node, then it is presumed to be under attack and is revoked. In CDD, node cooperation is also utilized along with node mobility for improving node capture attack detection. Two nodes that are within the communication range of each other exchange node information about the nodes that they are tracking. The two nodes compare the last meeting time of the set of nodes (common to each of them) that they are tracking and on the basis of the comparison of this time, it is inferred whether that particular node is under attack or not. So, CDD uses a more cooperative approach in comparison to SDD providing more efficient and reliable detection of the attack. Experimental results reveal that in CDD the detection time is much less (≈ 1.8 times) than that of SDD.

Defensive Mechanism: Both the works [34, 35] discussed below consider defense strategies for thwarting node capture attack in MWSNs. While [34] solely considers mechanism for defending node capture attack, [35] also considers other attacks such as DoS, message corruption etc.

More recently, in [34], Ren et al. proposed a scheme, Sensor Capture Resistance and Key Refreshing (SCARKER), for defending node capture attacks in MWSNs. It considers three issues while defending this attack- detecting node capture attack, preventing nodes that have recently joined the network in getting access to previous data, referred as forward security and discarding the possibility of the captured node to get hold of future network communication, known as backward security. The scheme is developed around a group key management scheme [36] and provides for backward and forward security in presence of node capture attack. It also ensures authentication and integrity during distribution of the group keys. An asymmetric architecture for MWSNs is considered here with sensor nodes having stringent computational capabilities and resources in comparison to the more resource sufficient

sink. During key updation of any group, each sensor node needs to perform one modular and one XOR operation. In the detection scheme, a node raises an alarm message if it finds that any other node is missing as every node is able to keep track of the existence of other nodes. The detection scheme is further divided into two classes: simple distributed detection, where a captured node is detected using local node information and cooperative distributed detection that takes the help of sensor nodes for improving the detection performance. The scheme is designed in such a manner that the time needed to compute the group key is directly proportional to the number of group members. The scheme is tested and evaluated in a testbed named as Sun SPOT [37]. The experiments conducted on the testbed are as follows—the time and energy required for computation of new group key, communication cost and storage cost in terms of the length of the secret key and the number of group members. Experimental results reveal that the scheme is very much suitable for MWSNs with minimum storage cost and affordable communication cost.

In [35], Mostarda et al. proposed a Distributed Intrusion Detection System (DIDS) for detecting and defending several attacks that MWSNs are exposed to. They have provided defense mechanisms against node capture attack. The basis of Intrusion Detection System (IDS) in this scheme is the use of a global automaton that is responsible for monitoring the messages entering and sent out from the network. The global automaton is inserted in filters and assigned to each sensor node. Every node is equipped with a set of filters that are responsible for performing specific tasks. Each filter is responsible for locally observing the node it resides on for validating the policy of the global automaton. Filters are implemented containing either the whole global automaton referred as heavy solution or parts of the global automaton known as light solution or intermediate solutions. If the filter detects an attack locally, then the message related to that node is discarded without raising any alarm. On the other hand, if the attack is on a larger scale, then alert message is sent towards the sink by the filter which detected the attack first. As each filter controls all the messages transmitted or received by the nodes, therefore if node capture attack takes place resulting in anomalous behaviour of the node, the filter is able to detect it. This work also deals in defending DoS attack by using a special property. It is considered that from an initial state to the final state a finite number of messages are to be received. If any node exceeds the number of invocations, it is inferred that the node is trying to launch DoS attack in the network.

3.1.2 Node Replication/Clone Attack

Node replication attack, also known as clone attack is launched by adversaries for producing node replicas in the network that are responsible of eavesdropping the messages communicated between nodes or to compromise the functions of the network. This section provides description of the works that deal with detection and defense of node replication attack in MWSNs.

Detection Mechanism: Existing node replication attack detection mechanisms proposed for MWSNs are described below.

Ho et al. [38] proposed a mechanism where the speed of the mobile node is observed to keep track whether node replication has occurred or not. It considers the fact that a mobile sensor node should not override the system-configured maximum speed. It is observed whether any node's speed is more than the maximum allowed speed. If so, then it is highly possible that there exist at least two nodes in the network with the same identity. On the basis of this, authors consider a statistical decision process Sequential Probability Ratio Test (SPRT) for their scheme. This decision process considers average number of observations to conclude whether attack has really taken place. SPRT can be considered as a one dimensional random walk having lower and upper limits that are associated with null and alternate hypotheses respectively. On the basis of any observation, a random walk starts from a point between two limits and then approaches the lower or upper limit. If the walk reaches or surpasses the lower or upper limit, it stops and the null or alternate hypothesis is selected accordingly. Every time a node changes its location, each of its neighbours asks for a signed claim containing the node's location and time information. The neighbour nodes decide probabilistically whether to transmit the claim to the sink. The sink is responsible for calculating the speed based on the information in the claim. When the maximum speed is crossed by the mobile node, it will lead the random walk to hit or cross the upper limit, thereby allowing the sink to accept the alternate hypothesis that the mobile node is replicated leading to revocation of the nodes. If maximum speed of the mobile node is not reached, the random walk will definitely hit or cross the lower limit, leading to the sink accepting the null hypothesis that mobile node has not been replicated. Authors extended their work in [39] considering more detailed analysis and different attack strategies of adversaries such as using game-theoretic approach etc.

Most recently, Conti et al. [40] proposed a solution for detecting clone attacks in MWSNs. Two protocols involving one-hop communication were proposed by them—History Information-exchange Protocol (HIP) and History Information-exchange Optimized Protocol (HOP) that differ in the amount of computation involved in their execution. The protocols involve node cooperation for replica detection and are executed in rounds. In each round the node stores a list in the form of log of the neighbour nodes met during a round along with their locations. Neighbouring nodes exchange their logs and every node compares these logs for determining whether in the same protocol round a particular node was present at more than one location. If such incident occurs, an alarm is generated throughout the network for revoking that particular node. In HOP protocol, the nodes compare logs not only of their one hop neighbours but also of other neighbour nodes to make the attack detection more effective. Also, the detection time in HOP is less than that of HIP though the number of computations is more in HOP than HIP. In both the protocols, nodes follow two procedures—spread and receive. In spread procedure each node is responsible for spreading its log information to neighbouring nodes. The receive procedure is used for receiving logs of other nodes followed by checking if inconsistency arises with respect to the location of nodes. Simulation results of their scheme demonstrate that the proposed protocols are very much efficient and effective in detecting clone/replication attacks.

Zhu et al. [41] addressed detection of node replication attacks in MWSNs considering the following- every node moves freely and randomly in the sensing region and meeting with other nodes are occasional and unpredictable. Authors used a lightweight token based authentication approach for detection purpose. In this approach the sink performs the task of periodically broadcasting a timestamp to the entire network and the timestamp is secured using a broadcast authentication protocol such as μ TESLA [2]. The broadcast is used for declaring the start of the detection round and also helps in synchronization among the sensor nodes. On receiving the timestamp, each node selects a secret seed which is either 0 or 1. The detection procedure comprises of two phases- token exchange phase and mutual authentication phase. During the token exchange phase when a node meets another node, a token is exchanged between them and both of them store the token. When the nodes meet again in the same detection round, they exchange their stored tokens and if the token matches then the nodes are authenticated. To reduce the overhead involved in token exchange, authors have devised a mechanism where the tokens are replaced by pairwise knowledge and the responses are replaced by commitments. Pairwise knowledge involves comparison done between stored timestamp of the last communication with the timestamp of the present communication. The authors claim their detection scheme to be efficient and requiring less overhead compared to other existing strategies.

In [42], Yu et al. proposed a scheme Efficient and Distributed Detection (EDD) for detecting node replication attack along with a variant scheme SEDD (Storage Efficient and Distributed Detection). Both EDD and SEDD are able to detect replication attack in a distributed manner without the involvement of the sink. In both the schemes, each node is capable of detecting this attack by itself. The attacked node is revoked by each node without using flooding. EDD is based on the philosophy that the number of times a node meets another node should be limited within a specific time interval. If the number of times one node is meeting another node with the same ID crosses a certain threshold in the given time interval, then the node is able to detect the presence of replica nodes in the network. Two steps are involved in implementation of EDD scheme. The first step known as off-line step is done by the network designer prior to deployment. Here the length of the time-interval and the setting of the threshold for differentiating between genuine and replica nodes is calculated. The next step is the on-line step that is performed by every node. In this step each node performs checking to confirm whether the encountered nodes are replicas or not by comparing the set threshold at the end of the time interval. To make the EDD scheme storage efficient SEDD is proposed based on the trade-off between storage overhead and the length of the time interval. Here, only a group of nodes is monitored by each node in a certain time interval instead of monitoring all the nodes. The group of monitored nodes is referred as monitor set. As only some nodes are monitored, the storage overhead per node is reduced to a great extent compared to EDD scheme. Authors claim that both the schemes are capable of identifying and detecting replica attacks with high accuracy and limited communication overhead.

Recently, Lou et al. [43] proposed a Single Hop Detection (SHD) distributed protocol for detecting replication attack in MWSNs. It is based on the fact that a node cannot be located at different positions in the network at the same time and

if this happens it indicates existence of replicas. Every node possesses its one-hop neighbour list. This protocol comprises of two phases i.e., the fingerprint claim and fingerprint verification. In the fingerprint claim phase, each node signs its neighbour's node list. The signed neighbour node list is referred as the fingerprint claim. The nodes then broadcast the fingerprint claim to their one-hop neighbourhood. When a node receives the fingerprint claim it runs an algorithm for taking the decision whether to act as the witness node for the claim node. If it accepts the fingerprint claim it first verifies it and then stores the same. In the fingerprint verification phase, two interacting nodes exchange their witnessed node lists and check whether there is any fingerprint claim conflict. If there is a conflict it can be inferred that replica nodes are present in the network.

Defensive Mechanism: The work given below provides defensive mechanism against node replication attack in MWSNs.

Yu et al. [44] devised a protocol eXtremely Efficient Detection (XED) for defending node replication attacks in MWSNs. Here the nodes are uniformly deployed and follow a particular model for moving. This defense mechanism is based on remember and challenge strategy, where, whenever any node meets any node, a random number is generated and stored by both the nodes. This is done so that if these two nodes meet once again they are able to ascertain whether they had met earlier by verifying the stored number. Each node uses a table to record the node ID, the generated random number and the received random number in their respective memory. If on random number request (when the two nodes meet again), one of them is not able to provide or provides a wrong random number, then the other node generates a replica node detection attack in the network. Based on probability calculations and the number of moves taken by the nodes, the proposed protocol is able to confirm the existence of replica nodes in the node. Authors claim that the advantages of their protocol lies in the fact that only constant communication cost is involved for replica attack detection and location information of nodes is not required.

3.1.3 Wormhole Attack

The adversary in wormhole attack records packets transmitted in one part of the network and tunnels to another place in the network for further retransmission. This section describes a work that provides defense against such attack in MWSNs.

Defensive Mechanism: A state-of-the-art wormhole defensive mechanism proposed for MWSNs is illustrated below.

Khalil et al. [45] provided defense against wormhole attack in mobile networks such as MWSNs. Here a secured Central Authority (CA) is used for tracking node locations globally while local monitoring is used for detecting malicious nodes as well as eliminating them locally. Sometimes the CA may also impose for global isolation of malicious nodes from the entire network when it suspects malicious nodes in large numbers. As this scheme uses a CA for tracking the mobile nodes and adversarial behaviour in the network, therefore, there is no need of any specialized hardware at each node for monitoring. The detection mechanism followed here is

of two types-local detection and global detection. In local detection the malicious node is detected in a distributed manner with the help of nodes monitoring locally referred as guards. For global detection, the CA is used where the CA gathers all the reports from the guards situated at different locations in the network. This scheme also proposes two protocols. The first protocol called Selfish Move Protocol (SMP) is based on the fact that a node can launch wormhole attack only if it forwards packets. Keeping this in mind, the mobile nodes are allowed to generate, send and receive only its own traffic but cannot forward traffic of other nodes. But SMP may lead to network disconnection if large number of nodes in the network become mobile. To address this issue, the second protocol called Connectivity Aided Protocol with Constant Velocity (CAP-CV) is proposed. The network disconnection problem is removed in CAP-CV protocol as it allows mobile nodes to also forward the packets. Simulation results of their scheme show that it is very much efficient in detecting and defending wormhole attacks in mobile networks.

3.2 State-of-the-Art Schemes

This section presents state-of-the-art works that have ensured the security services (Sect. 2.2) e.g., authentication, confidentiality etc. in MWSN and also secure key exchange.

Schmidt et al. [46] proposed security architecture for MWSNs that is capable of providing secure key exchange and ensure basic securities such as authentication. Their security architecture is based on three different phases: a pairwise key agreement phase for providing authentication and initial key exchange, establishing cluster creation for extending pairwise communication to broadcast inside the communication range and ensure authenticated and encrypted communication of data.

Pairwise key agreement is achieved using Blundo et al. scheme [47] that is based on a predistribution scheme in [48]. The predistribution scheme allows two nodes to obtain a pairwise secret that is exclusively shared among these two nodes. Also, it requires a certificate authority that is responsible for randomly generating a symmetric bivariate polynomial over a random finite field. The certificate authority is used for evaluating the polynomial where the coefficients of the polynomial are the key materials specific to each node. These key materials are transferred to the nodes in predeployed stage. The pairwise secret key is generated by two nodes using their private polynomial i.e. predeployed key materials. For establishing secure communication every node creates a randomly generated key within its neighbourhood. This random key is used by the node for encryption and authentication of its messages. This protocol allows a node to set up its key in a new network and also to gather knowledge about the keys of other nodes using only two messages in the form of request and reply. For encryption purpose the counter mode of operation [49] is used that enables message encryption without altering the length of the message. To

ensure integrity, Serpent algorithm [50] is used in this scheme that has good runtime behaviour as its implementation is possible using only logical operations.

Bairaktaris et al. [51] proposed a secured routing protocol that ensures integrity and confidentiality of the messages in MWSNs. Here the network is divided into a number of layers based on the hop distance of the nodes from the sink. To secure layer formation in the network, hash key chains are implemented. The whole key chain is known only to the sink. All messages exchanged are encrypted and encoded using one key of the key chain that is present in the nodes prior to deployment. Here secret shared key is generated using Elliptic curve model. The network area is divided into layers and sectors are uniformly distributed in the layers. Each sector designates a node as the sector head. Each sector head node generates a distributed depth-first traversal technique for creating a sector shared key using the elliptic curve model. Each node generates a random secret value with the help of which the sector head calculates its public key. Then this public key is sent in encrypted form to neighbouring nodes in the same layer. The neighbouring nodes that receive the public key repeat the same procedure for transmitting the public key further. Authors claim that their scheme is very much suitable for dynamically changing environments.

4 Open Issues

With the relentless proliferation of critical application areas e.g., clinical monitoring, wildlife monitoring etc. where mobile nodes can be used [52], researchers are facing new challenges for improvising security with severely limited node resources. In spite of the compendium of research activity and significant progress in the recent past [53], security in MWSNs has many open research issues which are still to be addressed. This section presents the open research areas in MWSNs relevant to security which are unexplored or those which can be improved by using certain techniques.

1. **Energy Consumption:** The problem of minimizing energy consumption of the security protocol deserves more attention. Even though, energy consumption issues are addressed in existing security protocols for MWSNs [35, 51], the energy efficiency goal still remains a challenge. Also, the network survivability will increase if nodes having heterogeneous resources are considered for deployment in MWSNs. Future work should focus on how heterogeneity can support mobility of nodes and allow implementation of resource conscious security methods.
2. **Design Complexity:** The node and sink mobility has great potential in improving the detection of adversaries [33] as well as optimize the energy consumption of the whole network [54]. Conti et al. [33] have shown that node mobility based security scheme has significantly less adversary detection time. However, very few researchers have exploited this issue. Since, mobile nodes are only capable of low-speed and short-distance mobility in real environment due to high power

consumption of locomotion, therefore, the moving trace of mobile nodes including sink must be optimized. It would be interesting to devise the distributed security schemes by leveraging node and/or sink mobility.

3. **Attack and Attack Models:** Providing security to any network is very much essential and MWSNs are no exception. But very few works have dealt with security in a very detailed manner. Although there are some works [33, 35] that have dealt with providing security against a handful number of active attacks prevalent in MWSN, still much work needs to be done for detection and defense against more number of existing active attacks e.g., message forgery attack. Much attention is also needed for providing defense against passive attacks e.g., eavesdropping in such networks. Also, a variety of attack models need to be explored to provide for more optimized solutions.
4. **Routing Security:** Routing is a prime area which needs special attention in MWSNs due to their inherent property of being mobile. It has been found that mobility plays a significant role in reducing common problems in MWSNs such as coverage, connectivity, energy hole problem [55]. In [51], the authors have explored the issue of secure routing protocol for MWSNs using hash key chains. Further, work can be done for developing schemes for efficient node discovery and effective transmission schedule for secure routing. Also, using QoS parameters as routing metrics will be beneficial for improving the functioning of existing secure routing algorithms. Also, nature inspired secure routing techniques can be explored for MWSNs.
5. **Application Area:** Mechanisms should be developed for MWSNs that ensure their applicability in new application areas. For example in industries, factory automation and process control are important aspects that demand continuous monitoring, envisaging the way for human-machine interaction via mobile devices [52]. Also several application areas such as search and rescue, environment monitoring applications etc. require cooperation between mobile nodes and fixed sensor nodes [56]. At the same time, security mechanism should be designed in such a manner that they have a wider exposure in many emerging application areas of MWSNs such as smart transport system for efficient management of traffic, social interaction etc.
6. **Intrusion Detection System:** In the context of IDS, more areas can be explored for developing security mechanisms that are to be used in the field of MWSNs. One such work [35] proposes an IDS the output of which generates a distributed secure routing protocol that can be used for MWSNs. Therefore, this area of MWSNs remains unexplored to quite an extent and can be an interesting area for future research.
7. **Key Management:** Key management is a crucial area when nodes in a network are mobile. A mobile node conducts key management by conducting source authentication with other nodes along its path of motion. This action by the mobile nodes requires additional energy as communication and processing are involved. The challenge for mobile nodes is performing the least number of source authentications. Such key management schemes are an essential part of MWSNs and calls for further research.

8. **Trust Management:** Trust management is a vital area of concern specially in mobile networks such as MWSNs due to frequent change in network topology. Trust management requires reputation based information from other nodes for helping mobile nodes to decide which nodes should be trusted and which should not be trusted. From the security point of view, trust management provides ample scope for future research in MWSNs.

5 Conclusion

MWSNs are emerging as an important research area due to their several advantages compared to static WSNs. Also, they can be implemented in wide range of application areas that are not much suitable for deployment with static WSNs. Similar to other networks, here also, security demands much attention. Some work has been done in the past with regard to MWSNs security but still many areas are left unexplored. Starting with, this chapter presents an insight into the different aspects of MWSNs such as their architecture, advantages, key issues and inherent challenges faced by the security schemes. This is followed by the works that have considered providing security in terms of basic securities and those which have addressed some of the attacks that are of frequent occurrence in MWSNs. Vivid discussions on the works that developed mechanisms for attack detection and attack mitigation are undertaken in this chapter. Finally, this chapter wraps up with the open research issues that can be considered in MWSNs with respect to security.

References

1. Sara, G.S., Sridharan, D.: Routing in mobile wireless sensor network: a survey. *Telecommun. Syst.* **57**(1), 51–79 (2014)
2. Anastasi, G., Conti, M., Francesco, M.D., Passarella, A.: Energy conservation in wireless sensor networks: a survey. *Ad Hoc Netw.* **7**(3), 537–568 (2009)
3. Chellappan, S., Bai, X., Ma, B., Xuan, D., Xu, C.: Mobility limited flip-based sensor networks deployment. *IEEE Trans. Parallel Distrib. Syst.* **18**(2), 199–211 (2007)
4. Yang, Y., Fonoage, M.I., Cardei, M.: Improving network lifetime with mobile wireless sensor networks. *Comput. Commun.* **33**(4), 409–419 (2010)
5. Dantu, K., Rahimi, M.H., Shah, H., Babel, S., Dhariwal, A., Sukhatme, G.S.: Robomote: enabling mobility in sensor networks. In: *Proceedings of 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, article no. 55 (2005)
6. Lee, U., Magistretti, E.O., Zhou, B.O., Gerla, M., Bellavista, P., Corradi, A.: Efficient data harvesting in mobile sensor platforms. In: *Proceedings of 4th PerCom Workshops*, pp. 352–356 (2006)
7. Munir, S.A., Ren, B., Jiao, W., Wang, B., Xie, D., Ma, J.: Mobile wireless sensor network: architecture and enabling technologies for ubiquitous computing. In: *Proceedings of 21st AINA Workshops*, vol. 2, pp. 113–120 (2007)
8. Ekici, E., Gu, Y., Bozdag, D.: Mobility-based communication in wireless sensor networks. *IEEE Commun. Mag.* **44**(7), 56–62 (2006)

9. Tilak, S., Kolar, V., Ghazaleh, N.B.A., Kang, K.D.: Dynamic localization control for mobile sensor networks. In: Proceedings of 24th IEEE International Performance, Computing and Communications Conference (IPCCC), pp. 587–592 (2005)
10. Kweon, K., Ghim, H., Hong, J., Yoon, H.: Grid-based energy efficient routing from multiple sources to multiple mobile sinks in wireless sensor networks. In: Proceedings of 4th International Conference on Wireless Pervasive Computing, pp. 185–189 (2009)
11. Yu, F., Park, S., Lee, E., Kim, S.H.: Elastic routing: a novel geographic routing for mobile sinks in wireless sensor networks. *IET Commun.* **4**(6), 716–727 (2010)
12. Mir, Z.H., Ko, Y.B.: A quadtree-based hierarchical data dissemination for mobile sensor networks. *Telecommun. Syst.* **36**(1–3), 117–128 (2007)
13. Cao, G., Kesidis, G., Porta, T.L., Yao, B., Phoha, S.: *Purposeful Mobility in Tactical Sensor Networks. Sensor Network Operations.* Wiley-IEEE Press (2006)
14. Wang, Y., Attebury, G., Ramamurthy, B.: A survey of security issues in wireless sensor networks. *IEEE Commun. Surv. Tutorials* **8**(2), 2–23 (2006)
15. Zhou, Y., Fang, Y., Zhang, Y.: Securing wireless sensor networks: a survey. *IEEE Commun. Surv. Tutorials* **10**(3), 6–28 (2008)
16. Chen, X., Makki, K., Yen, K., Pissinou, N.: Sensor network security: a survey. *IEEE Commun. Surv. Tutorials* **11**(2), 52–73 (2009)
17. Zin, S.M., Anuar, N.B., Kiah, M.L.M., Pathan, A.S.K.: Routing protocol design for secure wsn: review and open research issues. *J. Netw. Comput. Appl.* **41**, 517–530 (2014)
18. Xu, J., Zhu, W.T., Feng, D.G.: An efficient mutual authentication and key agreement protocol preserving user anonymity in mobile networks. *Comput. Commun.* **34**(3), 319–325 (2011)
19. Park, J.H.: An authentication protocol offering service anonymity of mobile device in ubiquitous environment. *J. Supercomputing* **62**(1), 105–117 (2012)
20. Wang, D., Wang, P., Liu, J.: Improved privacy-preserving authentication scheme for roaming service in mobile networks. In: Proceedings of 15th IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2014)
21. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L.S., Rubenstein, D.: Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebnet. *ACM Sigplan Not.* **37**(10), 96–107 (2002)
22. Friedman, J., Lee, D.C., Tsigkogiannis, I., Wong, S., Chao, D., Levin, D., Kaisera, W.J., Srivastava, M.B.: Ragobot: a new platform for wireless mobile sensor networks. In: Proceedings of International Conference on Distributed Computing in Sensor Systems (DCOSS), LNCS-3560, p. 412 (2005)
23. Shah, C.R., Roy, S., Jain, S., Brunette, W.: Data mules: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Netw.* **1**(2), 215–233 (2003)
24. Wang, G., Cao, G., Porta, T., Zhang, W.: Sensor relocation in mobile sensor networks. In: Proceedings of 24th IEEE INFOCOM, vol. 4, pp. 2302–2312 (2005)
25. Amundson, I., Koutsoukos, X., Sallai, J.: Mobile sensor localization and navigation using RF doppler shifts. In: Proceedings of 1st International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT), pp. 97–102 (2008)
26. Fang, L., Antsaklis, P.J., Montestruque, L., McMickell, M.B., Lemmon, M., Sun, Y., Fang, H., Koutroulis, I., Haenggi, M., Xie, M., Xie, X.: Design of a wireless assisted pedestrian dead reckoning system—the navmote experience. *IEEE Trans. Instrum. Measur.* **54**(6), 2342–2358 (2005)
27. Chen, C., Ma, J., Yu, K.: Designing energy efficient wireless sensor networks with mobile sinks. In: Proceedings of Workshop on World-Sensor-Web (WSW) at ACM SenSys, pp. 1–6 (2006)
28. Wang, Q., Hempstead, M., Yang, W.: A realistic power consumption model for wireless sensor network devices. In: Proceedings of 3rd IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON), vol. 1, pp. 286–295 (2006)
29. Natalizio, E., Loscri, V.: Controlled mobility in mobile sensor networks: advantages, issues and challenges. *Telecommun. Syst.* **52**(4), 2411–2418 (2013)

30. Kansal, A., Somasundara, A.A., Jea, D.D., Srivastava, M.B., Estrin, D.: Intelligent fluid infrastructure for embedded networks. In: Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 111–124 (2004)
31. Amundson, I., Koutsoukos, X.D.: A survey on localization for mobile wireless sensor networks. In: Proceedings of 2nd International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT), pp. 235–254 (2009)
32. Sakarindr, P., Ansari, N.: Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks. *IEEE Wirel. Commun.* **14**(5), 8–20 (2007)
33. Conti, M., Pietro, R.D., Mancini, L.V., Mei, A.: Emergent properties: detection of the node-capture attack in mobile wireless sensor networks. In: Proceedings of 1st ACM Conference on Wireless Network Security (WiSec), pp. 214–219 (2008)
34. Ren, Y., Oleshchuk, V., Li, F.Y., Sulisty, S.: SCARKER: a sensor capture resistance and key refreshing scheme for mobile WSNs. In: Proceedings of 36th IEEE International Conference on Local Computer Networks (LCN), pp. 271–274 (2011)
35. Mostarda, L., Navarra, A.: Distributed intrusion detection systems for enhancing security in mobile wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **4**(2), 83–109 (2008)
36. Zheng, X., Huang, C.T., Matthews, M.: In: Chinese remainder theorem based group key management. In: Proceedings of 45th ACM Annual Southeast Regional Conference (ACM-SE), pp. 266–271 (2007)
37. Sun Microsystems, Inc., Sun SPOT. <http://www.sunspotworld.com/>
38. Ho, J.W., Wright, M., Das, S.K.: Fast detection of replica node attacks in mobile sensor networks using sequential analysis. In: Proceedings of 28th IEEE INFOCOM, pp. 1773–1781 (2009)
39. Ho, J.W., Wright, M., Das, S.K.: Fast detection of mobile replica node attacks in wireless sensor networks using sequential hypothesis testing. *IEEE Trans. Mob. Comput.* **10**(6), 767–782 (2011)
40. Conti, M., Pietro, R.D., Spognardi, A.: Clone wars: distributed detection of clone attacks in mobile wsns. *J. Comput. Syst. Sci.* **80**(3), 654–669 (2014)
41. Zhu, W.T., Zhou, J., Deng, R.H., Bao, F.: Detecting node replication attacks in mobile sensor networks: theory and approaches. *Secur. Commun. Netw.* **5**(5), 496–507 (2012)
42. Yu, C.M., Lu, C.S., Kuo, S.Y.: Efficient and distributed detection of node replication attacks in mobile sensor networks. In: Proceedings of 70th International Vehicular Technology Conference Fall (VTC 2009-Fall), pp. 1–5 (2009)
43. Lou, Y., Zhang, Y., Liu, S.: Single hop detection of node clone attacks in mobile wireless sensor networks. In: Proceedings of International Workshop on Information and Electronics Engineering, PE-29, pp. 2798–2803 (2012)
44. Yu, C.M., Lu, C.S., Kuo, S.Y.: Mobile sensor network resilient against node replication attacks. In: Proceedings of 5th International Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 597–599 (2008)
45. Khalil, I., Bagchi, S., Shroff, N.B.: Mobiworp: mitigation of the wormhole attack in mobile multihop wireless networks. *Ad Hoc Netw.* **6**(3), 344–362 (2008)
46. Schmidt, S., Krahn, H., Fischer, S., Wätjen, D.: A security architecture for mobile wireless sensor networks. In: Proceedings of 1st European Workshop on Security in Ad-hoc and Sensor Networks (ESAS), LNCS-3313, pp. 166–177 (2005)
47. Blundo, C., Santis, A.D., Herzberg, A., Kutton, S., Vaccaro, U., Yung, M.: Perfectly-secure key distribution for dynamic conferences. In: Proceedings of 12th International Cryptology Conference (CRYPTO), LNCS-740, pp. 471–486 (1993)
48. Blom, R.: An optimal class of symmetric key generation systems. In: Proceedings of Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT), LNCS-209, pp. 335–338 (1985)
49. Dworkin, M.: NIST Special Publication 800–38A: Recommendation for Block Cipher Modes of Operation. <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>. (2001)
50. Anderson, R., Biham, E., Knudsen, L.: Serpent: A proposal for the advanced encryption standard. NIST AES Proposal **174** (1998)

51. Bairaktaris, K., Chatzigiannakis, I., Liagkou, V., Spirakis, P.G.: Adaptive probabilistic secure routing in mobile wireless sensor networks. In: Proceedings of 16th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 208–212 (2008)
52. Fotouhi, H., Zuniga, M., Alves, M., Koubaa, A., Marron, P.: Smart-HOP: a reliable handoff mechanism for mobile wireless sensor networks. In: Proceedings of 9th European Conference on Wireless Sensor Networks (EWSN), LNCS 7158, pp. 131–146 (2012)
53. Stavrou, E., Pitsillides, A., Hadjichristofi, G., Hadjicostis, C.: Security in future mobile sensor networks issues and challenges. In: Proceedings of International Conference on Security and Cryptography (SECRYPT), pp. 1–9 (2010)
54. Luo, J., Hubaux, J.P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In: Proceedings of 24th IEEE INFOCOM, pp. 1735–1746 (2005)
55. Wang, F., Liu, J.: Networked wireless sensor data collection: issues, challenges, and approaches. *IEEE Commun. Surv. Tutorials* **13**(4), 673–687 (2011)
56. Marron, P.J., Minder, D., Karnouskos, S.: *The Emerging Domain of Cooperating Objects: Definitions and Concepts*. Springer Publishing, New York (2012)

Dependable Communication for Mobile Robots in Industrial Wireless Mesh Networks

Timo Lindhorst and Edgar Nett

Abstract Mobile robots inevitably require a wireless network for communication. Wireless mesh networks (WMNs) allow for increased flexibility and an easy integration of mobile robots. Hence, they are well suited for present and future cyber-physical industrial applications. The high dependability requirements of that application domain, however, issue particular challenges to the communication network, especially regarding the mobility of stations. In this chapter, we present various approaches and mechanisms to provide dependable communication for mobile robots in industrial WMNs. These comprise different cross-layer techniques to allow for seamless mobility and an admission control that explicitly considers station mobility to avoid network overload. Evaluation results are gathered by both simulation and real-world experiments while case-studies supplement the results. The outcome is a network architecture that allows for seamless communication even for mission-critical applications on mobile robots.

1 Introduction

Mobile robots have enabled great potential and entered a wide field of applications. While stationary robots are settled in manufacturing processes since decades, mobile robots get more and more in focus of factory automation. Cyber-physical production systems [1] trigger a paradigm shift from centrally controlled manufacturing systems towards decentralized systems of cooperative, networked components. Intelligent manufacturing systems, storage systems, and mobile transport robots operate autonomously and exchange relevant information to ensure a flexible, highly adaptive, and self-optimizing production process. Mobile robots will work less statically

T. Lindhorst (✉) · E. Nett
Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
e-mail: lindhors@ivs.cs.ovgu.de

E. Nett
e-mail: nett@ivs.cs.uni-magdeburg.de

and individually, but dynamically perform tasks even in cooperation, thus pushing a suitable communication service into focus.

The envisioned flexible applications and the integration of mobile robots require the deployment of a wireless communication system that allows for ad-hoc communication and adapts to the actual environmental conditions and present demands of the cyber-physical applications. In contrast to other application domains, production processes require a highly dependable communication service that provides QoS-guarantees.

Wireless Mesh Networks (WMNs) [2, 3] are a suitable foundation to provide a flexible communication service. Instead of a wired backbone well known from common WLAN infrastructure networks, mesh routers span a fully wireless network, which provides self-configuration and self-healing capabilities by using a multi-hop routing protocol. Furthermore, the IEEE WLAN standard allows high data rates and thus allows data-intensive applications like video-based tele-operation of mobile robots. While various WMN routing protocols and mechanisms have been developed during the last decade, the latest version of the IEEE 802.11 standard [4] for the first time includes a mesh specification, which is expected to further boost the deployment of WMNs.

To provide the dependability required by industrial applications, however, several mechanisms are required to enhance the fundamental routing protocol. Due to the mobility of stations, routes continuously change. Links thereby may fail instantaneously. Suitable mechanism are thus required to timely detect link failures and avoid packet loss and thus allow *seamless mobility*, i.e. an uninterrupted communication even during movement of mobile stations.

A further foundation to ensure good performance and provide QoS guarantees is to avoid network overload, which otherwise would cause high latencies and packet losses. For this purpose, an admission control scheme can be implemented. This, however, requires a precise assessment of the network load induced by application's communications. The load thereby does not only depend on the amount of data the application sends, but it severely depends on the present network configuration. Due to the multi-hop relaying, the load may for instance double, if data has to be sent via two hops instead of one hop, as both transmissions share the same medium and thus interfere with each other. When mobile robots move within the network, the amount of network resources required for communication continuously changes, though. For dependable admissions the dynamics of the resource requirements thus need to be considered.

1.1 Contribution of the Book Chapter

The contribution of this chapter is a wireless mesh network architecture, which provides dependable communication for mobile robots in industrial environments. Hereby, we first present methods to ensure seamless mobility for mobile entities in Sect. 4. This is provided by two aspects: (1) A cross-layer link failure

detection (Sect. 4.2) and (2) the masking of packet losses within the network (Sect. 4.3). We present evaluation results from real-world experiments that are supplemented with a case study of a tele-operated mobile robot.

The contribution of Sect. 5 is an admission control scheme that explicitly considers mobility and hereby introduces support for mission-critical mobile applications. Besides another real-world case study, we provide evaluation results based on a simulation study.

2 Background

Communicating to mobile robots inevitably requires a wireless network. The probably most common wireless network architecture is the well-known infrastructure WLAN defined in the IEEE 802.11 standard [4]. While this requires a *wired* backbone network, in wireless mesh networks (WMNs) also the backbone is fully wireless, thus allowing for high flexibility and easy extensibility. A multi-hop routing protocol is used to find paths between stations that are not within direct communication range and in general provides self-organization and -healing capabilities. However, in contrast to mobile ad-hoc networks (MANETs), WMNs provide a backbone of stationary *mesh routers*. The applications run on *mesh clients*, which can be stationary or mobile.

There are two basic classes of routing protocols: *Pro-active* routing protocols maintain a representation of the whole network topology, whereas *reactive* protocols search for a path on demand. While the latter are mostly deployed in MANETs, we favor a pro-active routing protocol as it allows for instantaneous route availability and provides up-to-date topology information—the required communication overhead is acceptable as the envisioned networks in industrial environments are limited in size.

Our implementation is based on AWDS,¹ a proactive link-state-routing protocol operating on top of the link layer. Each station sends and receives HELLO-packets to detect its neighbors. If a certain number of HELLO-packets is received, the link is considered to be active. Each station broadcasts information about its present neighbors within the network. This allows all stations to reconstruct the entire network topology based on the information of the individual stations. Based on this information, each station creates a routing table for sending and forwarding packets on a multi-hop basis.

3 Related Work

The basis for our consideration of seamless mobility is the availability of sufficient network coverage, i.e. the availability of physical radio signal from the static backbone network. This can be achieved by a sophisticated planning of the placement of

¹<https://ivs-pm.ovgu.de/projects/awds>.

mesh routers like presented in [5]. However, even with the provision of sufficient radio coverage, seamless mobility requires further mechanisms above the physical layer.

Mobile stations frequently evoke packet losses when links fail as a station moves away. To avoid packet losses, a routing metric is required that timely detects reduced link quality and allows the routing protocol to switch to an alternative path. Cross-layer approaches show obvious performance benefits [6]. However, even works that try to reduce packet loss with a cross-layer metric still show loss rates of several percent [6–8]. Furthermore, several works address the prediction of link quality degradation to foster a seamless hand-off [9, 10]. However, while these works do improve the overall routing performance, they cannot prevent packet losses in case of almost instantaneous link failures.

In [11] Yackoski and Shen present an approach for link error recovery in MANETs to avoid losses. It extends the RTS/CTS mechanism of the standard IEEE 802.11 in a way that allows nearby stations to transiently take over the forwarding of a frame if the intended receiver does not respond. While this approach may reduce losses, it cannot ensure a reliable transmission.

In this chapter, we present an approach for seamless mobility in WMNs by combining two mechanisms: a fast cross-layer link failure detection and the masking of packet losses through local recovery measures, which totally avoid packet losses even while moving.

While there are various works regarding admission control in WMNs, mobility is rarely considered. We consider mobility as a severe challenge in admission control, as it strongly influences available resources and network load.

In [12] the authors propose an improvement to AODV to provide QoS in WMNs. They use a measurement-based approach to determine the present load in the network. By using a load-based routing metric, the performance of the network is improved in terms of higher throughput and lower latencies. However, no real guarantees can be provided and mobility is not considered.

In [13] an admission control for multipath routing is presented. The authors describe a sophisticated analytical model to estimate the available bandwidth based on properties like the physical bit rate, error rates, and back-off times utilizing parameters for the present contention window size. However, when determining the bandwidth requirement for new communications, they consider the same fixed values for all links in the network, e.g. a constant physical bit-rate of 2 MBit/s. They also only consider a static network when deciding about the acceptance of new communications and do not account for node mobility.

Hou et al. propose a routing scheme with bandwidth guarantees in [14]. They use information about available capacity on links and a clique-conflict-graph to model the impact of interfering transmissions to find the path that allows the highest possible throughput. However, neither the concept nor the evaluation considers the effect of mobility within the network.

The admission control scheme we present in this chapter will account for both variable physical bit rates and station mobility. By explicitly considering mobility, we introduce support for a new class of mission-critical applications.

4 Seamless Mobility

A basic aspect to provide dependable communication to mobile robots in a wireless network, is the requirement for seamless mobility: When a mobile robot is remote controlled within a production facility, the control must not be interrupted if the robot moves behind an obstacle. In emergency, even an autonomously moving robot must immediately react if an operator sends a stop command. A mobile station should continuously be connected to the network. However, the quality of a link may instantaneously decrease during the movement of the robot and the communication may be interrupted due to link failures. Thus, a timely detection of link failures is vital for a dependable communication service. For this purpose, we present a fast cross-layer link failure detection mechanism. While this mechanism allows to significantly decrease the number of packet losses in case of a link failure, it cannot totally avoid losses. Therefore, we introduce an additional mechanism to mask the remaining packet losses that are specific for wireless mobile applications. Hereby, the network is able to provide a reliable communication service to the application so that application designers does not have to cope with the peculiarities of wireless networks.

4.1 Link Failure Detection

In general, the detection of link failures is done within the routing protocol by means of *HELLO* messages: After the reception of a certain number of successive *HELLO* messages, the link is considered active and used for routing. If no more messages are received over a certain time interval, the link is considered to be broken. The detection delay can be optimized by tuning the parameters of the routing protocol, however, it can hardly be reduced down to a second and typically lies in the range of several seconds [15]. Thus, link failure detection with *HELLO* messages is obviously not sufficient to provide a dependable communication service for real-time applications.

In general, routing protocols also apply a routing metric to assign costs to individual links and find the shortest route to the destination. There is a great variety of routing metrics [6], which will eventually choose high costs for a failed link and thus cause a re-route. However, the process of detecting the increased costs, propagating it in the network, and re-route the traffic will as well take seconds.

4.2 Cross-Layer Accelerated Link Failure Detection

We propose a much faster detection of link failures that can be achieved by a cross-layer approach (x-layer accelerated link failure detection, XALF): Within the IEEE 802.11 MAC layer, retransmissions are used to increase the delivery probability

and mask sporadic transmission errors on the wireless medium. By evaluating information about transmission errors and delivery failures within the MAC layer, a link failure can be detected much faster than in the routing layer.

However, to ensure *correct* link failure detections, transient and permanent transmission errors must be distinguished. Transient errors should be compensated by retransmissions in the MAC layer, but if a link permanently fails, it can only be tackled in the routing layer by choosing an alternative route for communication. However, if a transient error is mis-detected as permanent, thus triggering a re-routing effort, the network topology is destabilized and additional overhead is incurred. Where further retransmissions would have been sufficient to deliver a frame, a link is now unavailable for routing and has to be re-established. Accordingly, these *false alerts* have to be avoided: An error should only be classified as permanent if it would be detected as a link failure at the routing layer as well.

Consider sending packets at a rate of 100 packets per second. A probability of false alerts of 0.01 % per packet would lead to incorrectly reporting a link failure every 100 s. To re-establish a link, further HELLO messages must be received which may take up to 10 s. In that case, even such a low rate of false alerts would cause the link to be down about 9 % of the time, thus vastly degrading the stability of the network topology. The negative effect to network performance caused by incorrect link failure decisions is also shown in [16, 17]. Hence, our goal is to achieve a fast link failure detection without compromising the stability of the network topology. Thus, false alerts have to be avoided by a reasonable model to distinguish transient and permanent errors.

In the following we present a classification model based on transmission errors using the basic bit rate. This has originally been published besides another one in [15]. Additionally, we proposed a data-mining-based approach to generate a classification model in [18].

4.2.1 Classification Based on Consecutive Transmission Errors

In general, on a failed link successive transmissions would fail. Thus, we consider consecutive transmission errors to detect link failures. However, when using information of transmission errors and retransmissions, we also have to consider the utilized data rates. Since the error probability for each transmission depends on the physical data rate, we cannot simply sum up consecutive frame transmission errors like in [16, 17, 19]. Instead, we only consider transmissions at the lowest (*basic*) data rate (according to the currently used modulation scheme). In general, rate adaption algorithms fall back to this rate if prior transmissions with higher data rates have failed. Only on errors at the basic data rate we can assume that the link actually has failed. Errors at higher rates may also occur if the algorithm has chosen the wrong rate. However, if a transmission is successful, independently of the used rate the link has obviously not failed.

For this model (*TxError*) we define a counter n_e , which counts the number of consecutive erroneous frame transmissions at the basic data rate. It is reset if

a transmission is successful either with the basic or any other data rate. However, if transmissions with other than the basic rate fail, the value remains unchanged. A link failure is assumed if $n_e \geq n_e^{thr}$ is true.

Appropriate values for n_e^{thr} correspond to considerations in [17, 19], however, only transmission errors with the basic data rate are considered. We performed an empirical model calibration using different setups with a supposedly stable link between two stations, tuning the parameter to a value that would not have triggered any false alerts. We varied the distance, the packet rate, and also the frequency band—details can be found in [15]. The calibration showed that in the 5 GHz band $n_e^{thr} \geq 3$ is an appropriate value to avoid false alerts, while in the 2.4 GHz band $n_e^{thr} \geq 5$ is suitable.

4.2.2 Evaluation of X-Layer Accelerated Link Failure Detection

We evaluate our XALF approach in a real-world WMN and compare it to a failure detection performed solely at the routing layer. For this purpose, we implemented an XALF module for the MadWifi² driver. This module comprises the *TxError* model with $n_e^{thr} = 4$, to employ the advantage regarding detecting link failures caused by mobility.

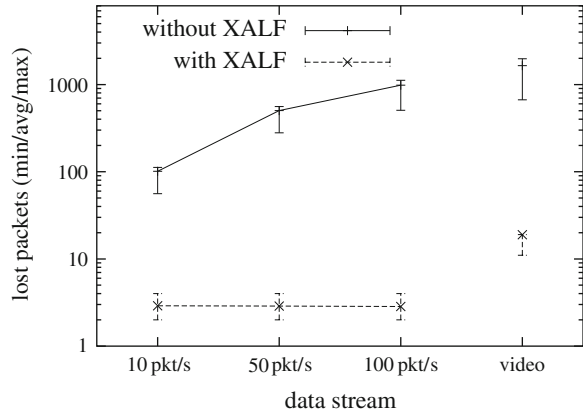
As first scenario we simulate a node failure. The sender S sends data to a receiver E using the route $S - R_1 - E$. We simulate a node failure on R_1 . When the link failure is detected, an alternate route $S - R_2 - R_3 - E$ is used to proceed communication. To simulate an instantaneous node failure, we just switch the WLAN card on node R_1 to a different frequency, thus interrupting its communication.

Different types of data streams were used for evaluation. Experiments have been performed with CPR data streams at 10, 50, and 100 packets per second, with 100 bytes per packet each. Furthermore, we simulated the transmission of a video stream. Here, a 35 kbyte UDP datagram is sent ten times per second. The datagram is fragmented into 19 packets, which thus become ready for transmission at almost the same time. We measured the number of lost packets on the receiving node E with our accelerated failure detection mechanism (XALF) disabled and enabled, totaling to eight different experiment scenarios. Each scenario has been repeated 200 times to achieve significant results.

Figure 1 shows the number of lost packets on a link failure with and without the XALF mechanism. It has to be noted that the y-axis has a logarithmic scale. It can be seen that the number of lost packets without XALF depends on the send rate and lies in the order of 100 up to 1000 packets. This is obvious, since the detection delay is bounded according to the parameters of the HELLO mechanism, which is $8 \times 1.4 \text{ s} = 11.2 \text{ s}$ in AWDS. As discussed at the beginning of Sect. 4.1 this delay can be reduced down to about one second with an acceptable overhead. However, when 100 packets/s are sent, still about 100 packets get lost on a link failure.

²<http://madwifi-project.de>.

Fig. 1 Lost packets on node failure



When XALF is used, the number of lost packets is independent of the packet rate. On average 3 packets get lost on a link failure, at most 4. Only when measuring the video stream, up to 19 packet losses can be observed. All 19 packets of one UDP datagram are passed to the network driver at almost the same time and stay in the network card's queue. Thus, the routing layer cannot determine an alternate route for the rest of the packets once a failure is detected. However, the loss of these 19 packets results in the loss of just one UDP datagram. From the application view, this means that only one video frame is lost.

Concluding from this, we can state that the number of packet losses on an instantaneous link failure can be reduced to 3 packets on average, independently of the used packet rate. The detection delay can thus be decreased from the order of a second to some milliseconds.

We additionally performed an evaluation where a link gradually degrades due to mobility. A node S sends data to a mobile robot E . Firstly, both are close to each other and a direct link is available. Then, the robot moves away and even around a corner along the corridor. Due to the increased distance the link $S - E$ fades out and finally fails. The communication can proceed over an alternate router $S - R - E$, once the link failure is detected.

Since this experiment could not be automated, we used only one stream of 50 packets per second (100 bytes per packet) and only performed 10 repetitions with and without XALF, respectively. Evaluating the results of this experiment we observed on average about 570 packet losses when XALF is not used. This is slightly more than in the previous scenario, which can be explained by single packet losses when the link quality gradually degrades. For the same reason, the number of packet losses slightly increases when deploying XALF. Here we achieve an average of 6, while minimum and maximum lies at 4 resp. 9. However, the average number of *consecutive* lost packets is even slightly less than in the previous scenario, since the rate adaption scheme increases the number of frame transmissions with the base rate when the

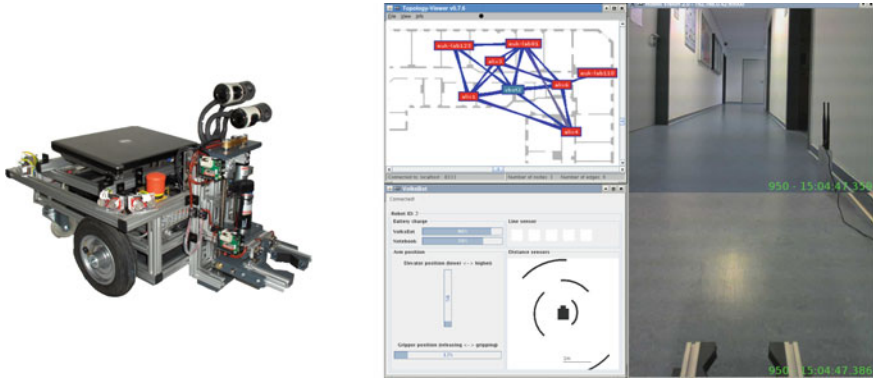


Fig. 2 Robot platform and operator view

link quality degrades. Thus, according to our model, a link failure is detected with less consecutive lost frames.

Concluding, we can state that our cross-layer accelerated link failure detection mechanism limits the number of lost packets independently of the send rate on instantaneous link failures as well as gradually degrading links caused by mobility. Thus we achieve an improvement of the detection time by two orders of magnitude.

4.2.3 Case Study: Tele-Operation of Mobile Robot

To prove the applicability of our approach we enhance the evaluation with a real-world case study.³ A mobile robot is controlled by an operator via a WMN deploying our cross-layer accelerated link failure detection. Steering commands from the operator are sent every 100 ms. Besides the values of 8 distance sensors, video streams of two attached cameras are transmitted with 10 frames per second in the opposite direction from the robot to the operator. The robot enters a fail-save state if it receives no control commands for 500 ms.

The robot and the operator view are depicted in Fig. 2. We deploy a mesh network in our office environment using dedicated mesh routers, a workstation PC is used for the operator. When trying to control the robot without XALF, the communication is interrupted when a mesh router is deactivated or the robot moves around a corner and leaves the communication range of its current neighbor. The robot enters its fail-state and the video is interrupted. After about 10s, the link failure is detected, the video stream resumes, and the robot can be controlled again.

Utilizing our XALF mechanism, critical communication interruptions are completely avoided. Neither when deactivating a router's power supply nor when moving within the area covered by the WMN the robot enters its fail-state. We could observe

³A video showing this case study can be found at: <https://euk.cs.ovgu.de/en/vbot-xalf>.

the loss of a single video frame, but the re-routing after a link failure detection has been fast enough to avoid the communication to fail. Thus, with XALF we can provide a dependable end-to-end communication service for a remote real-time control for a mobile robot.

4.3 Masking of Packet Losses

While the previously developed cross-layer link failure detection is able to minimize the number of packet losses in case of instantaneous link failures, it cannot totally avoid loss. In presence of mobility, link failures are common events and will thus cause a severe number of packet losses. Additionally, wireless communication is inherently interference-prone: both internal and external interference may result in sporadic packet losses. These causes for packet loss are non-existent in wired networks, but specific to wireless communication.

To cope with the modest reliability of wireless communication, higher layer protocols and applications often need to be adapted. For instance, to improve TCP performance in wireless networks, special mechanisms are required to distinguish congestion from interference-based losses [20]. Remotely controlling a robot requires sending control commands in a higher rate than actually needed by the control loop, as omission failures have to be tolerated. Thus, unlike intended by a layered protocol design, the properties of the underlying network are not transparent to the application.

The objective of this approach is to mask packet losses specific to wireless communication in WMNs without inducing severe latencies. Masking the inherent unreliability of the wireless medium eases protocol design for higher layers and applications.

4.3.1 Local Packet Recovery

Transmission failures are typically a subject of the link-layer, where two nodes exchange individual frames. When a frame cannot be delivered, this may be due to a transient fault like interference or a collision, or a permanent link failure may be the case. The latter induces a problem that we cannot solve on the link-layer: Since the destination node is not reachable anymore, every transmission attempt on the link will fail. Our approach is therefore to monitor each link for packet losses and, when a loss occurs, recover the packet in a form that allows a transmission over an alternative route.

In the following, we denote a packet a *lost packet*, when its corresponding transmission attempt on the link-layer has failed. The link-layer, passes such a packet back to the network layer via a new cross-layer interface—a process that we call *cross-layer recovery*. On the network layer, there is the possibility to switch a lost packet over a completely new route, which introduces the ability to tolerate packet losses caused by link failures. Using this strategy, we are able to exploit the redundancy

that is present in a WMN's topology to reliably deliver every single packet that is sent instead of just reacting to link failures after possibly tens of packets are lost.

We introduce a novel *transient re-routing* operation which is bound to the treatment of a single lost packet only. Forwarding tables will not be changed permanently, such that following packets will not take a new route through the network. In this way we avoid network instability since we do not alter link states during a transient re-routing operation.

We call the resulting concept *local packet recovery*, with the meaning of the term 'local' being twofold: First, we conduct recovery in a time-local manner, since transient re-routing will only affect one single packet, but not induce permanent changes in the network. Second, in contrast to an end-to-end acknowledgement mechanism, which provides reliability 'on top' between the source and destination nodes, our approach encounters faults right at the place where they occur in the network. This means, we re-send recovered packets from the intermediate node which detects the fault, saving both time and network load.

4.3.2 Transient Re-Routing

The re-routing operation has to be transient in the sense that the routing tables are not changed for packet delivery, as long as a link failure is not certain. While we originally developed three alternative approaches which are described in detail in [21], here, we focus on one approach.

As we consider mobility caused link failures to be one of the major sources for packet loss, we propose an approach to send a lost packet along an alternative route. This route does not contain the link on which the packet loss occurred, since we cannot safely determine the state of this link. To avoid network instability, the link is not permanently considered as failed and thus not removed from the topology. Instead, it is *marked* as unusable for the next forwarding attempt for the packet in question. As soon as a forwarding attempt for this packet succeeds, the mark is removed. Since we have access to the full topology information from the utilized link-state routing protocol, alternative routes can be calculated trivially.

Since we do not change the routing tables permanently, a special behavior has to be implemented on each router in order to conduct the switching along the bypass route. Like in traditional source routing approaches, we put the path information into the packet header, so all involved routers can forward the packet without having to do path calculations or having to alter their routing table.

4.3.3 Evaluation

The objective of our evaluation is to show that local packet recovery is able to mask all losses that are specific to wireless communication, i.e. sporadic losses and losses caused by link failure due to mobility. Additionally, we show that the latency

of packets that undergo fault recovery does not significantly differ from that of packets transmitted without any fault.

We implemented our concept into our AWDS routing protocol to evaluate it in a real-world testbed measuring packet round trip times (RTT). In the evaluation scenario, a mobile node communicates with a stationary node. Communication undergoes link failures due to mobility, as incident links to the mobile node fail permanently. Moreover, sporadic losses occur on every link. In this situation, we maintain statistics about the number and individual latencies of packets which happen to face faults during transmission and therefore would be omitted in the communication flow without local packet recovery. A constant flow of 100 packets per second, with 200bytes each packet, will be maintained during the test. The mobile node moves a defined path through an indoor hallway covered by 8 stationary nodes, increasing the distance to its communication partner and then approaching it again. Figure 3 shows the experiment setup, where the mobile node moved along the path $P_1 \rightarrow P_2 \rightarrow P_1 \rightarrow P_2 \rightarrow P_1$. The hop count from sender to receiver varies from one to three, with a maximum distance between the stations of about 40 m.

We execute one run without any fault handling (OFF) and one run with the described local packet recovery mechanism to mask packet losses (MASK). What should be seen in this experiment is that in the run employing local packet recovery no overall packet losses occur, i.e., all sporadic losses or failures of active links are masked within the network. Additionally, the packets that undergo fault handling should not incur significantly higher latencies than other packets. This means, the RTT distributions of both runs should have roughly the same shape.

During each run about 25,000 packets were send. Without masking losses, 210 packets (0.82 % of all sent packets) were lost. When applying local packet recovery, a similar percentage faced a fault (0.60 %), but all of them have been recovered

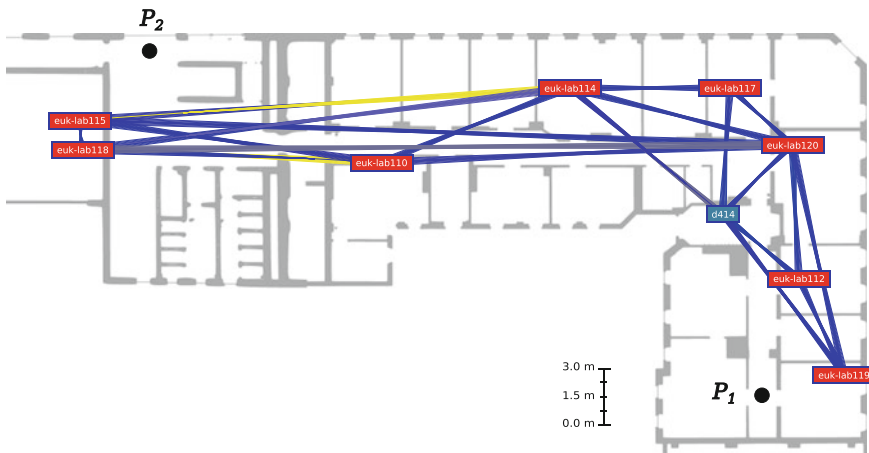


Fig. 3 Evaluation scenario. The mobile node d414 (on the right) communicates with the stationary node euk-lab115 (on the left) while moving between P_1 and P_2 with average walking speed

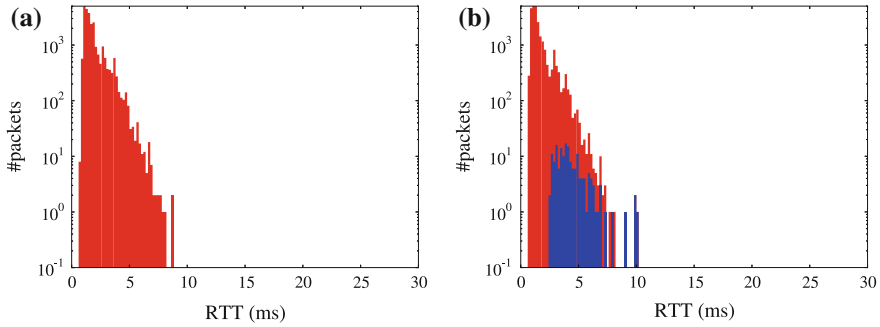


Fig. 4 Distribution of single RTT measurements. Measurements of marked packets are highlighted. **a** OFF. **b** MASK

using transient re-routing. Hence, the approach avoids every single potential packet loss in the communication flow.

Figure 4 shows the distributions of RTT measurements in both runs. Packets that undergo fault handling are marked, so they can be separated from the flow. The figure shows that most of the corresponding measurements, although they are obviously higher than those of most of the fault-free delivered packets, are located within the natural jitter of the measurements. We can say that the distribution of latencies does not change when local packet recovery is utilized.

5 Admission Control for Mobile Stations

While the previous mechanisms allow for seamless mobility by a fast detection of link failures and the masking of thereby caused packet losses, those mechanism cannot succeed in presence of network overload, as this would inevitably cause high latencies and packet losses. Thus, additionally, network overload has to be avoided to provide good performance and QoS guarantees to applications. This can be achieved by an appropriate admission control scheme. This, however, requires a precise assessment of the network utilization induced by application's communications.

A correct assessment of available network resources as well as the assessment of the resources required for a certain communication is already challenging in static wireless networks. Links with different physical bit rates (caused by different modulation techniques) require different amount of network resources to transmit the same amount of data. As the medium is shared, transmissions suffer from both external and self-interference. The latter is especially challenging in multi-hop wireless networks, as successive links on a route interfere in general and thus share the available capacity.

Mobile applications, i.e. applications running on mobile stations in the network, further exacerbate the problem of a correct resource assessment, as the mobility

causes dynamics regarding the required resources for transmissions to mobile stations. Thus, even though there might be enough resources to admit a new communication at the time it is requested, the movement of a station might cause increasing resource requirements that would eventually cause network overload. The reason for the varying resource requirements is twofold: First, the physical bit rate on a distinct link between two stations changes according to the present distance and environmental conditions. Second, if the route between sender and receiver changes, the required resources may increase due to self-interference caused by additional hops.

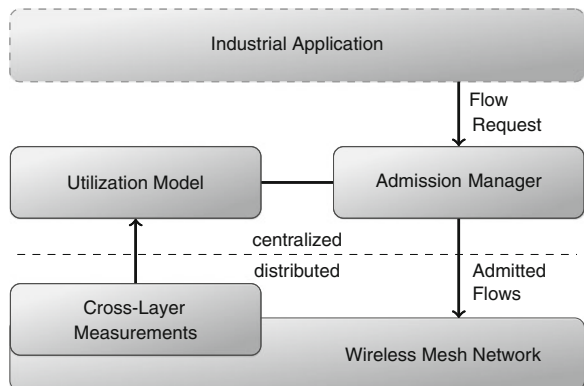
In this section, we first describe our admission control scheme and present a real-world case study that shows the effect of mobility to present network load conditions. Afterwards, we show how mobility is explicitly considered during resource reservations to extend the scheme to also allow mission-critical mobile applications.

5.1 Basic Admission Control Scheme

The admission manager scheme is shown in Fig. 5. It is described in detail in [22]. In the following, we shortly summarize the main aspects.

For an admission manager, the resource to be managed is the utilization of the wireless medium. As it is shared among neighbored stations, transmissions of different nodes add up to the present utilization. Thus, the central admission manager (CAM) uses a utilization model to decide about network capacity and network load induced by new communication flows: Based on cross-layer measurements, the present utilization is determined in terms of the medium time that is consumed by ongoing transmissions. Whenever a new communication flow f is requested, the utilization induced by this flow is determined by the present network topology, the route between the sender and the receiver of this flow, as well as the link qualities obtained from cross-layer measurements on the particular route as $U_f = R_f \times C_f$, where R_f is the

Fig. 5 Feedback loop for admission control



packet rate of the flow and $C_f = \sum c_i$ is the sum of the particular costs (in terms of expected transmit time [22]) per link on the route.

The overall capacity of the network is modeled as one global resource that is shared by all transmissions in the network, the overall utilization is thus obtained as $U = \sum U_f$. This is a pessimistic simplification as sufficient distant stations can transmit simultaneously. However, in the envisioned limited-scale networks, this may only result in a low overestimation of the present and induced load. While this may sacrifice some efficiency, it is uncritical regarding our objective of a dependable admission control scheme.

If the utilization exceeds a certain level, packet losses and increased latencies occur due to network overload [23]. Thus, the admission manager has to prevent over-utilization. Hereby, certain resource reserves are kept to be tolerant against faults induced by network dynamics and allow local recovery mechanisms in the network without taking risk of temporary medium overload ($U \leq U_{max} < 1$). As further measure, the admission manager may revoke previously admitted flows to avoid the violation of guarantees of remaining flows. Revoked flows may be re-requested by the application and are granted if present utilization allows it.

5.2 Real-World Case Study

While we did an empirical evaluation of our admission control scheme, which can be found in [22], we also run a real-world case study to show the applicability of our approach. The setup equals the one described in Sect. 4.2.3: A mobile robot is remotely tele-operated sending steering commands from an operator to the robot while streaming back live video from the robot to the operator. In this case, we utilized our admission scheme to explicitly request communication flows. Besides the flow for control commands (16 Kbit/s) and the flow for the video (6000 Kbit/s), we request 4 background flows between some other stations in the network, however with a lower priority. The admission manager tracked the utilization caused by all flows during the runtime of the experiment.

Figure 6 shows the utilization as monitored by the CAM. On the x-scale, the real time of the experiment (minutes and seconds) is depicted. The y-scale shows the total utilization caused by all flows. The admission threshold (70%) and the maximum allowed utilization (80%) are displayed as red horizontal lines.

At 27:35, the four background flows (*bg #1.4*) are initialized. At 27:54, the tele-operation console is activated (*control* and *video*), displacing *bg #1* after a short moment. The *control* flow however has such a low utilization (0.2% at its highest point) that it is not noticeable in the graph.

After its activation, the robot moves through the operational area, increasing the distance to the operator and thereby increasing the capacity demand gradually. At 29:10, its demand is so high that only a single background stream (*bg #1*) can be tolerated.

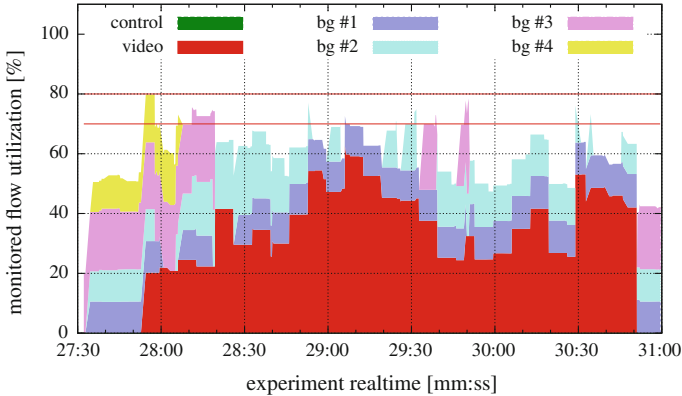


Fig. 6 Flows and utilization

A broad description of various further measurement results regarding this case study can be found in [24]. Here, we want to emphasize, that the provision of QoS by avoiding network overload during the operation of the robot is only possible, if enough network resources are available. In this case study, background flows could be canceled to free additional resources for the increasing capacity demand of the flow to the mobile robot. However, to also handle mobility in mission-critical applications, the dynamic resource requirements has to be considered at time of flow request.

5.3 Considering Mission-Critical Applications

In industrial environments there are also mission-critical applications, i.e. applications, that expect an admitted communication to last for the entire runtime of the application. The variation of resource requirements that was shown in the previous case study, however, poses a challenge to admission control for mission-critical applications: If an increase of resource consumption caused by mobile applications leads to network overload, we cannot revoke previous admissions to prevent this.

To provide a dependable admission control that must not revoke admissions for mission-critical applications, the admission management has to consider the dynamics caused by mobile stations. It is not sufficient to only consider the present situation when QoS communications are initially requested. Instead, it is important to ensure that the provided guarantees can also be maintained when mobile stations move within the operational area causing the network to change.

We define the following terminology:

- **Best-effort (BE) flow:** A flow that might be revoked by the admission manager after its admission at any time in order to avoid network overload.
- **Mission-critical (MC) flow:** A flow that, once admitted, must not be revoked by the admission manager. Only the application can release the flow.

Applications design defines which type of flow to request in accordance to the upper characteristics. MC should be chosen over BE only if necessary.

Considering MC flows, an admission scheme has to foresee increasing capacity demands caused by station mobility. Thus, for the CAM it is not sufficient to only consider the present state of the network when a new flow is requested at time t_0 . Instead, it should only admit new flows if the utilization is expected to stay below the limit for the entire time the flow may be active:

$$U(t) = \sum_i U_{f_i}(t) \leq U_{max} \quad \forall t > t_0 \quad (1)$$

whereas $\hat{U}_{f_i}(t)$ is the utilization induced by a flow f_i at an instant of time in the future.

We try to find an upper bound \hat{U}_f for the utilization that is expected to be reached during the life-time of a flow f . By the reservation of sufficient resources to handle this upper bound of the utilization for the flow, we allow the provision of QoS guarantees even when mobile stations move and thus cause increasing capacity demands.

To reserve sufficient resources for MC admissions in presence of mobility, it is important that the upper bound for the overall utilization of all MC flows stays below the defined limit. Thus, whenever a new MC flow is requested, the admission manager needs to ensure

$$\hat{U} = \sum_f \hat{U}_f < U_{max} \quad (2)$$

In contrast to Eq. 1, these estimates are time invariant and may overestimate the overall utilization, as not all flows reach their maximum at the same time.

We developed two approaches to determine an upper-bound utilization to be considered for mission-critical applications. While a detailed description can be found in [25], we describe the main ideas of both approaches in the following.

- **Worst case: longest possible path:** As worst case, a mobile station may move to a place that requires the flow to be send via the *longest* path in the network, i.e. the path that requires the most resources. As we know the present network topology, we calculate the maximum utilization that would be required in that case and consider it as upper bound.
- **Maximum utilization observed in the past:** Considering mobile industrial applications, we assume recurrent mobility patterns, e.g. a robot that carries goods to a machine or products from a machine to a warehouse. We name these distinct applications running on mobile platforms a *job*, which also comprise a requests of a distinct communication flow. We assume that such a job can be identified by a job id, which is passed as additional parameter during flow requests. The admission manager monitors the maximum utilization of subsequent application runs with the same job id and maintains a list of the observed values. Whenever a new flow is requested, the maximum value of the previously monitored flows for this job id is considered as upper bound for the present request. Regarding the first flow of a

job we fall back to the previous worst-case approach or may alternatively, during an uncritical learning phase, accept it if present conditions allow it.

The first approach is pessimistic but thus also very reliable. However, in most cases the upper bound of utilization considered on flow requests is much higher than eventually required. The second approach tries to be more efficient regarding resource reservation thus allowing a better overall network utilization.

5.4 Evaluation

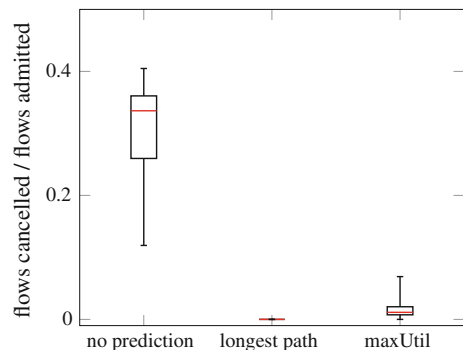
We implemented both approaches in our admission scheme. However, for the sake of reproducibility, the evaluation is performed using the network simulator ns-3.

To compare the two approaches with each other and with the original non-predictive scheme, a series of artificially generated scenarios is used. For comparability, in each scenario all variants are evaluated with the same initial conditions. The stationary routers are placed in a square grid topology of 3×3 routers. Ten mobile nodes, which operate within this area, are initially placed randomly. One of the stationary nodes also functions as the CAM.

The mobile nodes are controlled by an application that mimics the behavior of a transport robot in industrial environments. When a mobile node is idle, it randomly chooses one of 10 randomly generated jobs, which describes a certain movement pattern in the operation area and further specifies the parameters of a flow that is required to be available during execution of the job. The bandwidth required for the jobs lay in the range between 1600 and 3000 Kbit/s. The bandwidth and the number of mobile nodes have been chosen to generate a sufficient load for the evaluation while having only a minimum number of mobile nodes idling due to limited capacity.

We compare the proposed approaches regarding the dependability of the admission of MC flows, i.e. the ability of the admission control mechanism to maintain admissions even in presence of station mobility. We simulated 20 different scenarios for 30 min each and recorded how often a previously admitted flow is cancelled. Figure 7 shows for each variant the number of cancellations relative to the number

Fig. 7 Ratio of flows cancelled to flows admitted



of flows admitted. The line in the middle represents the median, the box the first and third quartile and the whiskers the minimum and maximum over all simulation runs. For comparison, we also show the results for the case that no prediction is used.

As expected, when the CAM does not consider the future utilization at the time of a request, it has to cancel between 12 % and 40 % of the previously admitted flows later on. This includes events where a flow has to be cancelled multiple times before a mobile station is able to complete its job. On average 63 of 204 admitted flows had to be cancelled, what shows that MC applications cannot be handled if mobility is not considered.

When using the estimation based on the longest path, the CAM never cancels previously admitted flows, because it overestimates the utilization caused by the flow. However, as a consequence, only 33 flows have been admitted on average in each scenario.

When using the maximum utilization of previous jobs to predict the utilization of a flow, in some scenarios the CAM never had to cancel an admitted flow. In other scenarios up to 7 % of all admitted flows had to be terminated. On average 88 flows have been admitted and in more than half of the scenarios only up to one flow had to be terminated. If a flow had to be terminated, most of the time it happened in the first few minutes of that simulation run. This is to be expected because there is no information about previous jobs. When a station requests a flow that has not been previously seen, for this evaluation the CAM uses the optimistic approach and admits it if the present utilization allows it. If later on the utilization exceeds the maximum allowed value, one of the active flows will be cancelled. While in a real setup we consider BE flows to be present and could be cancelled to compensate the increased resource requirements, the evaluation only comprised MC flows.

The results show that our scheme to explicitly handle mobility within admission management allows to also support mission-critical applications: the variation of the resource requirements caused by moving stations is considered on time of flow request, hence, dependable communication for mobile robots can be provided. Additional evaluations in [25] further show that the network can be fully utilized, if both MC and BE flows are present.

6 Conclusion

In this chapter, we presented a wireless mesh network architecture that addresses various aspects to provide dependable communication for mobile robots in industrial environments. To ensure seamless mobility we first developed a cross-layer mechanism that allows a fast detection of link failures without sacrificing network stability. Additionally, we presented an approach for local packet recovery and transient re-routing to mask the remaining packet losses without incurring significant latency. Finally, we extended an admission control scheme to explicitly account for station mobility, thus introducing support for the new class of mission-critical applications to mobile stations. Real-world evaluations and case studies show the applicability of our architecture.

References

1. Lee, E.A.: Cyber physical systems: design challenges. In: 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), May 2008
2. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. *Comput. Netw.* **47**(4), 445–487 (2005)
3. Benyamina, D., Hafid, A., Gendreau, M.: Wireless mesh networks design—a survey. *Commun. IEEE Surv. Tutorials* **14**(2) (2012)
4. IEEE Computer Society. Standard 802.11—Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, (ISO/IEC 8802–11) edition (2012)
5. Ivanov, S., Nett, E.: Wireless Mesh Networks, chapter Achieving Fault-tolerant Network Topology in Wireless Mesh Networks. Intech (2012)
6. Campista, M.E.M., Esposito, P.M., Moraes, I.M., Costa, L.H.M., Duarte, O.C.M., Passos, D.G., de Albuquerque, C.V.N., Saade, D.C.M., Rubinstein M.G.: Routing metrics and protocols for wireless mesh networks. *IEEE Netw.* **22**(1), 6–12 (2008)
7. Esposito, P.M., Campista, M.E.M., Moraes, I.M., Costa, L.H., Duarte, O., Rubinstein, M.G.: Implementing the expected transmission time metric for OLSR wireless mesh networks. In: *Wireless Days* (2008)
8. Passos, D., Albuquerque, C., Campista, M., Costa, L.K., Duarte, O.B.: Minimum loss multiplicative routing metrics for wireless mesh networks. *J. Internet Serv. Appl.* **1**(3), 201–214 (2011)
9. Farkas, K., Hossmann, T., Legendre, F., Plattner, B., Das, S.K.: Link quality prediction in mesh networks. *Comput. Commun.* **31**(8), 1497–1512 (2008)
10. Wanalertlak, W., Lee, B., Chansu, Y., Kim, M., Park, Seung-Min, Kim, Won-Tae: Behavior-based mobility prediction for seamless handoffs in mobile wireless networks. *Wirel. Netw.* **17**(3), 645–658 (2011)
11. Yackoski, J., Shen, C.-C.: Cross-layer inference-based fast link error recovery for manets. In: *IEEE Wireless Communications and Networking Conference*, vol. 2, pp. 715–722 (2006)
12. Liu, L., Zhu, L., Lin, L., Wu, Q.: Improvement of AODV routing protocol with QoS support in wireless mesh networks. In: *International Conference on Solid State Devices and Materials Science*, Macao (2012)
13. Zhao, P., Yang, X., Wang, J., Liu, B., Wang, J.: Admission control on multipath routing in 802.11-based wireless mesh networks. *Ad Hoc Netw.* **11**(8), 2235–2251 (2013)
14. Hou, R., Lui, K., Baker, F., Li, J.: Hop-by-hop routing in wireless mesh networks with bandwidth guarantees. *IEEE Trans. Mobile Comput.* **11**(2), 264–277 (2012)
15. Lindhorst, T., Lukas, G., Nett, E.: Modeling fast link failure detection for dependable wireless mesh networks. In: *IEEE International Symposium on Network Computing and Applications*, IEEE, pp. 44–51, July 2010
16. Marandin, D.: Performance evaluation of failed link detection in mobile ad hoc networks. In: *Proceedings of the 3rd Mediterranean Ad Hoc networking Conference (MedHoc Net)*, June 2004
17. Owada, Y., Maeno, T., Imai, H., Mase, K.: OLSRv2 implementation and performance evaluation with link layer feedback. In: *IWCMC '07 Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing*, pp. 67–72. New York, USA, ACM (2007)
18. Lindhorst, T., Lukas, G., Nett, E., Mock, M.: Data-mining-based link failure detection for wireless mesh networks. In: *IEEE International Symposium on Reliable Distributed Systems*, IEEE, pp. 353–357, Nov 2010
19. Velayos, H., Karlsson, G.: Techniques to reduce IEEE 802.11b handoff time. In: *Proceedings of IEEE International Conference on Communications*, June 2004
20. Balakrishnan, H., Padmanabhan, V.N., Seshan, S., Katz, R.H.: A comparison of mechanisms for improving tcp performance over wireless links. *Netw. IEEE/ACM Trans.* **5**(6), 756–769 (1997)

21. Engelhardt, F., Lindhorst, T., Nett, E.: Tolerating packet losses in wireless mesh networks. In: 28th IEEE International Parallel and Distributed Processing Symposium, IEEE Computer Society, pp. 1470–1479 (2013)
22. Lukas, G., Lindhorst, T., Nett, E.: Modeling medium utilization for admission control in industrial wireless mesh networks. In: IEEE International Symposium on Reliable Distributed Systems, pp. 65–74 (2011)
23. Jardosh, A.P., Ramachandran, K.N., Almeroth, K.C., Belding-Royer, E.M.: Understanding congestion in IEEE 802.11b wireless networks. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, IMC'05, pp. 25–25, Berkeley, CA, USA, USENIX Association (2005)
24. Lindhorst, T., Lukas, G., Nett, E.: Wireless mesh network infrastructure for industrial applications—a case study of tele-operated mobile robots. In: 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (2013)
25. Lindhorst, T., Weseloh, B., Nett, E.: Dependable admission control for mission-critical mobile applications in wireless mesh networks. In: Proceedings of the 33rd IEEE Symposium on Reliable Distributed Systems, submitted (2014)

Part IV

Mobility

Automatic Merging of Vehicles: Design, Algorithms, Performance

Gurulingesh Raravi, Vipul Shingde and Krithi Ramamritham

Abstract Automakers are trying to make vehicles more intelligent and safe by embedding processors which can be used to implement “by-wire” applications for taking smart decisions on the road or assisting the driver in doing the same. Given this proliferation, there is a need to minimize the computational capacity required without affecting the performance and safety of the applications. These applications have stringent requirements on data freshness and completion time of the tasks. Our work studies one such safety-related application, Automatic Merge Control (AMC), which ensures safe vehicle maneuver in the region where n roads intersect. As our contributions, we (i) propose three algorithms for AMC and analyze their behavior assuming single-lane roads and vehicles that allow AMC to control their behavior, (ii) enhance AMC to provide solution for multiple-lane road scenarios and also accommodate mixed traffic (both AMC-controlled and human-driven vehicles), (iii) demonstrate how Dedicated Short Range Communication based wireless communication protocol can be leveraged for the development of AMC and (iv) present a real-time approach towards designing AMC by integrating mode-change and real-time repository concepts for reducing the processing requirements. Simulations and a prototype implementation on robotic platforms demonstrate the advantages of our approach for constructing AMC systems.

Keywords Automatic merge control · Cyber-physical systems · DSRC communication · Intelligent transportation systems

G. Raravi—The author is currently at Xerox Research Centre India.

G. Raravi (✉)
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
e-mail: guhri@isep.ipp.pt

V. Shingde · K. Ramamritham
Indian Institute of Technology Bombay, Mumbai, India
e-mail: vipulgs@cse.iitb.ac.in

K. Ramamritham
e-mail: krithi@cse.iitb.ac.in

1 Introduction

Safety-critical automotive applications such as drive-by-wire, are distributed and real-time in nature. The current practice of implementing each application as an independent black-box excludes any possibility of sharing the microprocessors. This trend of increasing the number of microprocessors in relation to individual applications will be difficult to maintain both in terms of cost and integration complexity. Hence, the microprocessors must be effectively used to make fully electronically controlled vehicles a reality. Also, all these safety-related systems have stringent timing requirements apart from having specific functional requirements. Hence, it is necessary to provide real-time guarantees for such systems.

With a larger goal of understanding the requirements of by-wire applications and hence a fully autonomous vehicle, in this paper, we study one such safety related application, namely Automatic Merge Control (AMC), which ensures safe vehicle maneuvering in the region where two or more roads intersect (referred to as *merge region*). This application is distributed in nature, requiring vehicles to communicate with each other and take a collective decision to ensure safety of all the relevant vehicles. It ensures that the time instants at which two vehicles cross the merge region are separated by at least δ (which depends on the area of the merge region and permitted velocity of vehicles in the region) to ensure safety, by giving commands to adapt their velocities appropriately. AMC has three sub-problems: (S1) Ensure safe separation distance between vehicles; (S2) Ensure safe maneuvering of vehicles at merge region and determine the Merge Sequence (MS), i.e., the order in which vehicles cross the merge region; and (S3) Minimize the time taken by vehicles to cross the merge region, e.g., average DTTI. DTTI of a vehicle is the time it takes to cross the merge region from *Area of Interest* (AoI) (see Sect. 2).

Figure 1a shows an intersection of two roads namely, $Road_1$ and $Road_2$ (each having a single lane), where vehicles on each road are denoted as x_{ij} and they are at least S distance apart from each other. It is assumed that $Road_i$ contains m_i vehicles, where $1 \leq i \leq 2$ (road index) and $1 \leq j \leq m_i$ (vehicle index). For simplifying the

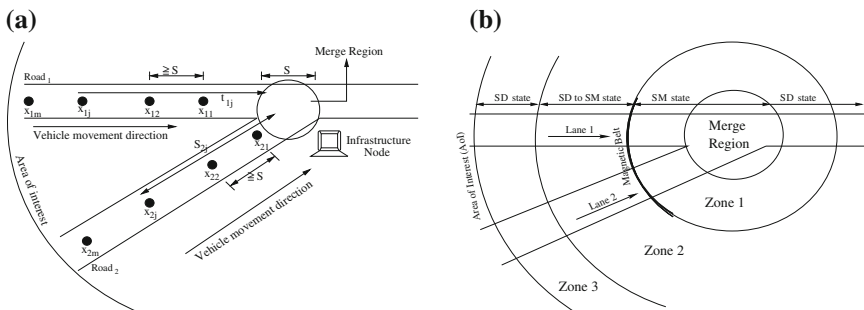


Fig. 1 Automatic merge control system and overview of our design approach. **a** Automatic merge control system. **b** Zone-based partitioning of AoI

vehicles' *profile* (velocity, acceleration, etc.) computations (see Sect. 3.4), we have assumed that the diameter of the merge region is equal to 'S'.

Our goal is to develop algorithms and provide system support for AMC with the following objectives:

1. Maintain a safe separation distance between vehicles on each lane and ensure their safe maneuvering at intersections (referred to as merge regions).
2. Minimize Driving-Time-To-Intersection (DTTI)—the time taken by a vehicle to cross merge region.
3. Efficiently utilize the computational resources.
4. Minimize average duration of *external control*, i.e., the time during which vehicles are controlled by AMC system—particularly important in a case where vehicles' *profiles* (velocity, acceleration, etc.) are decided by an external entity (i.e., by AMC algorithms).
5. Provide real-time support so that tasks meet their deadlines to ensure safety of the system.

Utilizing the basic framework of [1], this paper, (i) presents an approach for determining the profile of vehicles, (ii) presents HoL_{PE} algorithm that handles high traffic densities, (iii) relaxes assumptions of single-lane roads and necessity of all vehicles being AMC-controlled, (iv) addresses communication failure scenarios and (v) provides simulation and experimental results on vehicular robots to demonstrate the benefits of our approach for developing an AMC system.

In the paper, the term *road* implies a road with a single lane unless specified otherwise. Hence, the terms *lane* and *road* are used interchangeably till Sect. 4 where we relax the assumption of roads having a single lane.

The rest of this paper is organized as follows. Section 2 gives an overview of our approach and design. New algorithms (under certain assumptions) along with a mechanism to determine profiles of vehicles are proposed in Sect. 3. The relaxation of some of these assumptions is discussed in Sect. 4. Section 5 discusses our real-time system support and inter-vehicle communication mechanism for AMC. Experimental setup and results are discussed in Sects. 6 and 7 respectively. Section 8 presents the related work followed by conclusions.

2 Overview of Our Approach

Our algorithms determine the safe merging sequence and profile to be followed by every vehicle to achieve safe maneuvering at intersections. Among our three algorithms, Head of the Lane (HoL) is a distributed solution that considers only the head/lead vehicle on each lane for determining the merge sequence (MS). *Head/lead vehicle* is defined as the vehicle closest to the merge region and whose profile is not yet decided by AMC algorithm. Head of the Lane with Propagation Effect (HoL_{PE}) is similar to HoL, but while inserting a head vehicle in the MS, it considers its effect on the average DTTI of vehicles that are behind the head vehicles. All Feasible

Sequences (AFS) is a centralized solution which takes the current *snapshot of vehicles* (certain number of vehicles) with their profiles and determines the MS and new profiles of all the vehicles in the snapshot at once.

Our system design exploits two well known design techniques from real-time domain namely, *mode change protocol* [2] and *real-time data repository* [3]. Both approaches lead to effective utilization of the CPU capacity by understanding task and data characteristics. The mode-change protocol is a *task-centric* approach that allows the designer to vary the task sets and their characteristics (e.g., period) as the system moves from one mode to another. The real-time data repository model is a *data-centric* approach that decides the task characteristics given the freshness requirements of the data. We have integrated both these approaches to leverage the advantages offered by both the methods. The Dedicated Short Range Communication (DSRC) based wireless communication protocol used for inter-vehicle communication is explained in Sect. 5.2.

To tackle the earlier mentioned sub-problems, S1 and S2, we have abstracted AMC to operate in one of the two states, (i) safe distance (SD) state where vehicles maintain safe separation distance from each other, or (ii) safe merge (SM) state where safe merging of vehicles is ensured. To draw a boundary on the section of road that comes under AMC and to easily understand the system behavior, we have defined an *Area of Interest* (AoI) which is divided into three zones as shown in Fig. 1b:

- **Zone3** (Z_3): The vehicles enter this zone with random profiles but they maintain a safe separation distance from their immediately following vehicles, i.e., the system operates in SD state in this zone. Even though vehicles are required to operate in SD state in this zone, they are not directly under the control of AMC system, as each vehicle itself determines its own profile and follows it.
- **Zone2** (Z_2): The vehicles present here are considered by AMC algorithms for determining the merge sequence and vehicles' profiles. In this zone, the system transitions from SD to SM state. The exact time of transition of each vehicle from SD to SM state depends on the merge algorithm.
- **Zone1** (Z_1): The vehicles enter this zone with new profiles (set by AMC algorithm in Z_2) and are never reconsidered for profile computation. The system operates in SM state here. As soon as a vehicle crosses the merge region, it switches to SD state.

This zone-based partitioning has the following benefits:

1. **Stability:** The vehicles being left undisturbed in Z_1 add to the stability of the system, as these vehicles are very close to the merge region and have little flexibility to adapt to any changes in their profile.
2. **Continuous Stream of Vehicles:** This zonal partitioning also helps algorithms to deal with continuous stream of vehicles as described in Sect. 3.
3. **Modular Design:** This has helped to design the system in two exclusive states namely, SD and SM thereby allowing us to look at different sub-problems in different states.

Having summarized our approach, we now present our algorithms along with a method for determining profiles. We make the following assumptions while solving the problem. All the vehicles run the same algorithm and each vehicle follows its own profile decided by the algorithm and during this course, no vehicle overtakes the vehicle in-front of it (referred to as its *leading vehicle*) and further no breakdown of vehicles or accidents occur. Each road has a single lane (which is relaxed in Sect. 4) and AMC deals with only those vehicles that are in AoI.

3 AMC Algorithms

In this section, we present our algorithms and also describe how to determine the profiles of vehicles.

3.1 Head of Lane (HoL) Algorithm

This algorithm is a distributed solution that determines the merge sequence in an iterative manner. This approach is inspired by the way drivers resolve conflict at an intersection in practice. The drivers who are closest to the merge region on each lane decide among themselves the order in which they pass through the merge region. This algorithm achieves the goal of safe maneuvering by considering the lead vehicle (referred to as Head of the Lane (HoL) vehicle) on each lane for determining the MS. This approach postpones taking over the autonomy from vehicles as long as possible and easily maps to the way merging happen in real-world scenarios where vehicles are not automated.

The algorithmic steps are listed below:

1. When a lead vehicle on any of the roads reaches magnetic belt, it declares itself as HoL of that road, e.g., x_{11} on $Lane_1$ in Fig. 1a. This event triggers HoL to be “elected” for the other road ($Lane_2$).
2. The vehicle which is nearest to the merge region on $Lane_2$ and whose profile is not yet determined by AMC is elected as HoL, e.g., x_{21} in Fig. 1a. (If this road is empty then x_{11} elects itself as the winner, and gets inserted into the MS; Goto step 1.)
3. Now, x_{21} sends its profile to x_{11} which performs the computation to determine the winner and a new profile that the winner has to follow till it crosses the merge region. Both the sequences, i.e., x_{11} followed by x_{21} (x_{11}, x_{21}) and vice versa (x_{21}, x_{11}) are considered while making the decision. If only one sequence is *feasible* (see Sect. 3.4), then it is chosen else the sequence with minimum average DTTI is chosen. Head/Lead vehicle of the chosen sequence is declared as winner.
4. x_{11} sends the computed profile of the winner to x_{21} and the winner is inserted into the MS. The algorithm also considers the profile of the vehicle that was inserted

into the MS in the previous iteration so as to maintain the safe distance from that vehicle and also to enter the merge region only after that vehicle exits. If x_{11} is not the winner in step 3, it repeats steps 2–4 by declaring itself as the leader on $Lane_1$ till it gets inserted into the MS.

5. Steps 1–4 are repeated in a continuous loop.

When x_{11} is decided as the winner in step-4, the other option for electing the new HoL on that road (instead of waiting till one of the vehicles reaches the magnetic belt in step-1) is to immediately declare x_{12} as the new HoL. We decided to take the current option since it allows vehicles to enjoy the autonomous state as long as possible without compromising on the safety. The algorithm is sporadically executed whenever a vehicle reaches the magnetic belt.

The fact that HoL algorithm considers only two (head) vehicles at a time (for computation) makes it easy to deploy in real-world. But, it might fail to work under high traffic densities since it does not consider the effect of its decision on the DTTI of vehicles that are behind the lead vehicles. In other words, the local decision made might affect the global scenario (simulation results in Sect. 7 confirm this). To overcome this drawback, we propose two other algorithms, HoL_{PE} and AFS.

3.2 Head of Lane with Propagation Effect (HoL_{PE}) Algorithm

HoL algorithm is greedy in nature w.r.t. DTTI, i.e., it takes the decision on the basis of the information at hand (profiles of two head vehicles) and makes the choice that seems best for the moment (one with the minimum DTTI) without worrying about the effect that this decision may have in future (average DTTI of all the vehicles in Z_2). HoL_{PE} is an enhanced version of HoL algorithm that tries to incorporate this feature, i.e., it makes the decision about present HoL vehicles by considering its effect on the DTTI of the vehicles that are behind them. In other words, HoL_{PE} tries to consider the effect of allowing a particular head vehicle (from the available two) first, on the vehicles behind by measuring the DTTI delay introduced in the system. Incorporating this feature into HoL_{PE} helps it to operate under higher traffic densities compared to HoL (discussed in experimental results in Sect. 7).

The algorithmic steps of HoL_{PE} are exactly same as those for HoL algorithm shown in Sect. 3.1 except for step-3. In Sect. 3.1, only one vehicle from the other lane communicates its profile whereas in this case:

- Besides x_{21} , all other vehicles in Z_2 ($x_{12}, x_{1m}, x_{22} \cdots x_{2r}$) also communicate their profiles.
- Both the sequences, i.e., x_{11} followed by x_{21} and vice versa are considered while making the decision. If only one sequence is feasible (see Sect. 3.4), that sequence is chosen. If both sequences (x_{11}, x_{21}) and (x_{21}, x_{11}) are feasible then for each sequence, the algorithm computes the profiles of other vehicles in Z_2 (and hence the average DTTI) considering two possible merge sequences that might result in future, one for each lane, i.e.,

- For sequence (x_{11}, x_{21}) , both $(x_{11}, x_{21}, x_{12} \cdots x_{1m})$ and $(x_{11}, x_{21}, x_{22} \cdots x_{2r})$ are considered and $DTTI1_{AVG}$ is calculated.
- For sequence (x_{21}, x_{11}) , both $(x_{21}, x_{11}, x_{12} \cdots x_{1m})$ and $(x_{21}, x_{11}, x_{22} \cdots x_{2r})$ are considered and $DTTI2_{AVG}$ is calculated.
- $\min(DTTI1_{AVG}, DTTI2_{AVG})$ is computed and the head vehicle associated with that sequence is chosen as the winner and is inserted into the MS.

3.3 All Feasible Sequences (AFS) Algorithm

This algorithm assumes that an intelligent (communication + computation capable) *infrastructure node* is situated near the merge region. Such a node performs all the computations and determines profiles of the vehicles and it does this without altering the profiles of vehicles whose order in the sequence is already decided. This algorithm is a centralized solution that considers all the relevant vehicles from a snapshot, i.e., all the vehicles in Z_2 , at once and determines the MS and profiles of these vehicles. The algorithm is initiated by a new vehicle sending the *MergeInitiate* message upon reaching the magnetic belt on one of the roads. The infrastructure node collects the profiles of all the vehicles present in Z_2 and examines all possible combinations to determine the new profiles. The newly computed profiles are communicated back to the vehicles. Since this algorithm works with a snapshot of vehicles, to make this algorithm work for continuous stream of vehicles, we need to consider:

1. **Frequency of invocation of algorithm:** The algorithm is invoked when one of the lanes contains all new vehicles (i.e., vehicles whose profiles are not yet computed) in Z_2 . The algorithm could be run more frequently, i.e., when $N \geq 1$ vehicles enter Z_2 . But this will increase the computational overhead since majority of the vehicles in new snapshot will be from old snapshot (whose profiles are already computed) which leads to redundant computations and also frequent changes in vehicles profiles.
2. **Handling previous vehicles:** Whenever the algorithm is executed again with completely new vehicles on one of the lanes (say *Lane*₁), other lane (*Lane*₂) might have old vehicles (vehicles considered in previous snapshot) in Z_2 , whose profiles have already been computed in previous iteration. Apart from all new vehicles in Z_2 , only those vehicles (from previous snapshot) whose profiles will be affected are reconsidered for the current iteration. Such vehicles are identified as follows. Minimum DTTI (say, $DTTI_{1,min}$) of the first new vehicle on the lane which triggered the current iteration (*Lane*₁) is determined with the help of its current profile and profile of vehicle immediately in front of it. Only those old vehicles from the other lane (*Lane*₂) whose DTTI (determined using the profile computed in previous iteration) is greater than $DTTI_{1,min}$ are reconsidered.

Table 1 shows the comparison of the AMC algorithms where n is the number of vehicles in Z_2 . We now present a procedure to determine vehicles' profiles.

Table 1 Comparison of AMC algorithms presented in this work

Algorithm	AFS	HoL	HoL _{PE}
Nature	Centralized	Distributed	Distributed
(#vehicles considered, #profiles decided) per iteration	(n, n)	(2,1)	(n,1)
External control duration	High	Low	Low
Performance under high traffic density	Good	Poor	Average
Network traffic	Bursty	Non-bursty	Bursty

3.4 Profile Determination

Let $d_{ij}(t)$ and $f_{ij}(t)$ of vehicle x_{ij} denote its distance from merge region and its profile as a function of time t . Profile of a vehicle to be inserted in the MS is determined by considering the profiles of vehicles that are already in MS. There are three possibilities: (i) MS is empty, (ii) Last inserted vehicle in MS is from the same road and (iii) Last inserted vehicle in MS is from a different road.

Here, we assume that profiles of vehicles that have already been inserted in MS are known. Let F_{ij} denote the set of feasible profiles for vehicle x_{ij} which satisfy the safety criterion. We need to pick up a profile from this set which yields the lowest value of DTTI for this vehicle and will also help to minimize DTTI for vehicles that are going to be inserted into MS after x_{ij} . While determining the profile of a vehicle that is to be inserted in the MS, the following four constraints (imposed by the system) should never be violated: (a) the maximum allowed velocity for vehicles (b) the maximum allowed acceleration for vehicles (c) safe distance of separation (DoS) from its leading vehicle on the same road and (d) safe DoS from the vehicle that was previously inserted in the MS. Here, restriction (d) is the same as that of (c) when the previously inserted vehicle in the MS is the leading vehicle (from the same road) of the vehicle under consideration. Restriction (d) needs special attention only when the previously inserted vehicle is from a different road than the vehicle under consideration.

The goal is to find the best possible (i.e., optimal) profile for each vehicle so that their DTTI is minimized. Below we have formulated the constraints for each of the three cases. Let $t_{ij,SM}$, $t_{ij,ME}$ and $t_{ij,MX}$ denote the time at which vehicle x_{ij} switches to SM state, Enters and exits the merge region respectively.

1. MS is empty: In this case, the computation of the vehicle's profile to be inserted into the merge sequence depends only on restrictions (a) and (b), i.e.:

$$0 \leq -d'_{ij}(t) \leq V_{max} \quad \forall t \in [t_{ij,SM}, t_{ij,MX}] \quad (1)$$

$$A_{min} \leq d''_{ij}(t) \leq A_{max} \quad \forall t \in [t_{ij,SM}, t_{ij,MX}] \quad (2)$$

where d' and d'' denote first and second derivative of function $d(t)$ respectively (i.e., velocity and acceleration) and V_{max} , A_{min} and A_{max} represent the maximum

velocity, and the minimum and maximum acceleration, respectively. $d()$ represents distance of the vehicle from merge region, which is decreasing w.r.t. time and hence the velocity of vehicle, i.e., $d'()$ has a negative sign to it in Eq.(1). The optimal profile for the vehicle in this case is determined as follows: accelerate the vehicle with $a = A_{max}$, till its velocity reaches V_{max} and move with this velocity till it exits the merge region.

2. Last inserted vehicle in the MS is from the same road: In this case, algorithm needs to ensure that it satisfies restriction (c) (of maintaining the safe DoS from its leading vehicle which is nothing but the previously inserted vehicle in the MS) as well, apart from (a) and (b). Let the last inserted vehicle in MS be $x_{i(j-1)}$. Restriction (c) can be expressed as:

$$d_{ij}(t) \geq d_{i(j-1)}(t) + S \quad \forall t \in [t_{ij,SM}, t_{i(j-1),MX}] \tag{3}$$

where S is the desired safe DoS between vehicles. Initially (i.e., at $t = t_{ij,SM}$), the distance between vehicles $x_{i(j-1)}$ and x_{ij} would be at least S . Let the extra distance, if any, be $S_{ij,i(j-1)}$, where

$$S_{ij,i(j-1)} = d_{ij}(t_{ij,SM}) - d_{i(j-1)}(t_{ij,SM}) - S \tag{4}$$

In order to achieve optimal (minimum) DTTI, the algorithm should try to determine the profile of x_{ij} so that this additional distance/gap reduces to zero as soon as possible and once it is achieved, the vehicle x_{ij} should travel at the same velocity as $x_{i(j-1)}$ so as to maintain safe DoS. While determining such a profile we need to respect constraints (a) and (b) mentioned above. Based on the initial DoS and the constraints under which the vehicles operate, we need to follow one of the methods shown in Fig. 2a, b to determine the profile.

Figure 2a depicts a scenario when the initial DoS is large and hence constraints on both maximum acceleration as well as velocity need to be considered whereas Fig. 2b depicts a scenario where initial DoS is small and hence vehicle can reduce the additional gap to 0 without reaching the maximum speed and hence this scenario considers only the constraint on maximum acceleration. If the initial separation distance ($S_{ij,i(j-1)} + S$) between x_{ij} and $x_{i(j-1)}$ is very large then the profile described

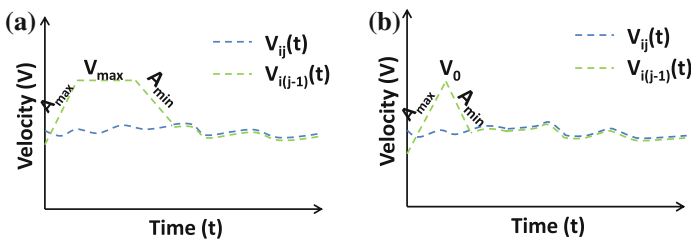


Fig. 2 Two scenarios to be considered while determining the vehicle profile when the last inserted vehicle in the sequence is from the same road. **a** Scenario1: DoS is large. **b** Scenario2: DoS is small

for an empty merge sequence will have to be applied. Note that the area under these curves representing the distance covered by vehicle x_{ij} must be equal to $S_{ij,i(j-1)} + S$.

The detailed analysis is omitted here and can be found in Appendix of [4].

3. Last inserted vehicle in the MS is from a different road: In this case, all the four restrictions mentioned earlier need to be satisfied while determining the profile of the vehicle under consideration. Here, since the leading vehicle is different from the previously inserted vehicle in the MS, it is essential to ensure that safe distance is maintained not only from its leading vehicle but also from the previously inserted vehicle in the MS. By doing this, the algorithm ensures that there is no possibility of (i) collision between the vehicle under consideration and its leading vehicle and (ii) two vehicles from different roads being present in the merge region at the same time. In other words, it ensures that if the two consecutive vehicles inserted in the MS are from different roads, then the second vehicle enters the merge region only after the first one exits. This restriction (4) can be expressed as:

$$d_{ij}(t) = d_{kl}(t) + S \quad \forall t \in [t_{kl,ME}, t_{kl,MX}] \quad (5)$$

Note the timing window considered by Eqs. (3) and (5). Equation (3) ensures that safe distance from the leading vehicle is maintained from the time the vehicle under consideration switches to SM state till it exits the merge region. Equation (5) ensures that safe distance is maintained from the previously inserted vehicle of a different road from the time it enters the merge region till it exits the merge region.

Initially, optimal solution (say, $d_{ij,opt}$) for x_{ij} can be computed using only Eqs. (1)–(3). Then $d_{ij,opt}$ can be extended to meet Eq. (5) by providing deceleration to prevent multiple vehicles from different roads being present in the merge region at same time.

The details on computing acceleration of vehicles for determining the profiles can be found in Appendix of [4].

4 Relaxation of Assumptions

We now propose methods to relax the assumptions of single-lane roads and necessity of all vehicles being AMC-controlled to tackle real-world scenarios.

4.1 Multiple-Lane Roads

AMC can be easily extended to multiple-lane road-merging scenarios where only innermost lanes of two intersecting roads merge into a single lane (and the rest of the lanes continue to exist after merge region). For example, a road with 4 lanes and a road with 3 lanes merge together resulting in a road with 6 lanes, as the innermost lanes of both roads merge into a single lane. The only vehicles that AMC system

needs to consider are the ones on the innermost lanes, as vehicles only on these lanes might collide with one another. Here, our AMC algorithms can be applied without any modifications by imposing a restriction that vehicles cannot change lanes (in or out of innermost lane) once they enter Z_2 .

The scenario where all the m lanes from different roads merge into a single lane rarely arises in real-world and hence detailed description of it is not provided. However, the algorithms naturally extend to such a scenario—only thing that changes is the number of vehicles that the algorithms have to consider at any given point of time to make a decision. For example, HoL has to consider m vehicles at a time (as opposed to just two) for deciding single vehicle's profile.

4.2 AMC-Controlled and Non-AMC-Controlled Vehicles

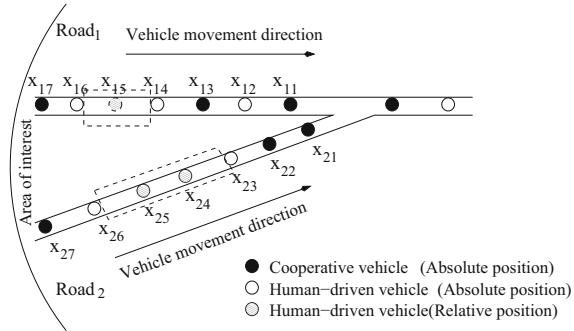
A sizable fraction of vehicles are human-driven (referred to as non-AMC-controlled vehicles) in real-world. To handle such a mix, we have made a practically-reasonable assumption that non-AMC-controlled vehicles do not display erratic behavior. In particular, they give preference to their own as well as neighboring vehicles safety while changing their course, speed etc.

Let Vehicle Mixture (VM) ratio be α . Here, VM ratio refers to the fraction of total vehicles that are non-AMC-controlled. There is a trade-off involved between designing a system that handles higher VM ratio and the system's ability to minimize average DTTI—a system that handles large VM ratio is likely to perform poorly with respect to achieving low average DTTI.

Since some of the vehicles are non-AMC-controlled, system cannot communicate with these vehicles and hence cannot directly enforce them to follow profiles as in earlier case where all the vehicles were AMC-controlled. One way to handle this scenario is by using the concept of virtual vehicles [5]. *Virtual vehicle* is generated by mapping a vehicle (which is at a distance d from the merge region) from one lane onto another lane (at a distance d from the merge region), and the following vehicle on the other lane tries to maintain safe distance from this virtual vehicle. We now discuss few scenarios involving mixed vehicles and corresponding solutions.

Both Single-Lane Roads Consider a single-lane merge scenario as shown in Fig. 3. The current profile of a non-AMC-controlled vehicle can be communicated to the AMC system if it is immediately being followed by an AMC-controlled vehicle. Such non-AMC-controlled vehicles can then be mapped onto another lane and be used as virtual vehicles. But note that we still cannot force a non-AMC-controlled vehicle to maintain safety distance from a virtual vehicle. In such scenarios the behavior of non-AMC-controlled vehicle can be controlled by restricting non-AMC-controlled vehicle to be inserted in MS only after AMC-controlled or non-AMC-controlled vehicle from the same road. Hence, for the scenario shown in Fig. 3, vehicle from *Road*₂ should not be merged between x_{11} and x_{12} . Thus, we can define an *encapsulation* of vehicles i.e., group of vehicles in which the first and the last

Fig. 3 Vehicle map for single lane merge scenario



vehicle must be AMC-controlled and the vehicles in between can be either AMC-controlled or non-AMC-controlled for each non-AMC-controlled vehicle, and use this encapsulation to ensure safety of vehicles. An encapsulation is an atomic entity and hence it has to cross the merge region in its entirety.

The mixed-traffic algorithm is listed below:

1. **Vehicle map and/or encapsulation generation:** Some mechanism such as magnetic belt at Z_3 boundary helps AMC system (or infrastructure node) to know whether the vehicle that entered Z_3 is AMC-controlled or non-AMC-controlled. AMC-controlled vehicle sends its current profile and that of its leading vehicle to the infrastructure node as soon as it enters Z_3 whereas non-AMC-controlled vehicle fails to do so. Figure 3 shows a single-lane road merge scenario with mixed traffic along with the generated vehicle map.

As can be seen, exact position of few vehicles (referred to as *absolute position*) is known to the infrastructure node, while for the other vehicles, it knows only the area in which they lie (referred to as *relative position*) shown by dotted line. In other words, the absolute position of only those non-AMC-controlled vehicles (apart from AMC-controlled vehicles) is known that are either preceded or succeeded by AMC-controlled vehicle as AMC-controlled vehicles can pass on that information to infrastructure node. For all other non-AMC-controlled vehicles, only their relative position can be estimated considering that they will maintain safe distance from the other vehicles.

2. **Merge order determination:** Merge order is determined periodically using the following algorithm:
 - (a) When a AMC-controlled vehicle reaches merge region (referred to as *initiator*), it sends a request to the infrastructure node to determine the merge order.
 - (b) *Infrastructure node* asks HoL vehicles on both roads, and the vehicle most recently inserted in the MS to communicate their profile.
 - (c) These vehicles communicate their profile along with non-AMC-controlled vehicle(s) profile (if any) directly adjacent to each of them. Note that HoL vehicles are elected in the same way as described in Sect. 3.1.

- (d) *Infrastructure node* then checks whether the last inserted vehicle (LI_{veh}) in the MS and the HoL vehicle on the other road are valid for becoming a *virtual vehicle* (vv) for the *initiator*. LI_{veh} is valid if it is not from the same road as the *initiator* and HoL vehicle from the other road is valid if it is tail of an encapsulation.
- if** (both are valid) **then**
 if (*initiator* is head of the encapsulation) **then**
 it chooses the vehicle nearer to it as its vv
 else it chooses the last inserted vehicle as vv
 end if
else if (the LI_{veh} is valid) **then** it chooses the LI_{veh} as its vv
else No vv is assigned to it
end if
- (e) **if** (*initiator* is head of an encapsulation) **then**
 insert all vehicles in this encapsulation in MS
else insert only *initiator* in MS
end if
 Inform *initiator* the vv (if any) assigned to it.
- (f) Algorithm terminates temporarily.

Both Multiple-lane Roads A multiple-lane road scenario where only innermost lanes merge can be handled by imposing a restriction that only AMC-controlled vehicles are allowed in innermost lanes, restricting all the non-AMC-controlled vehicles to outer lanes. As only AMC-controlled vehicles are present in the merging lanes the AMC system design proposed in Sect. 4.1 can be directly used here.

Though AMC problem is applicable to merging of two or more roads, we end this section by mentioning that it is a generic problem and our solutions can be applied to scenarios like Highway-Ramp, Lane Closures, etc.

5 System Support for AMC

We now discuss (i) real-time support for reducing the processing requirements without missing the deadlines and (ii) inter-vehicle communication support.

5.1 Real-Time Support: Dual-Mode, Two-Level Data Repository Approach

This section describes how the AMC system integrates mode-change and real-time data repository protocols.

Mode-change protocol: Real-time applications typically exhibit mutually exclusive phases/modes of operation and control [2]. A mode change will typically lead

to either: adding/deleting a task or increasing/decreasing the execution time of a task or increasing/decreasing the frequency of execution of a task. In different modes, we can have the sensing tasks execute at different frequencies to deal with dynamically varying data and we can have different set of tasks active in different modes. Hence, we do not need to have all the tasks active at all the time. AMC is designed with this approach and has two mutually exclusive modes of operation:

- **Non-Critical (NC) Mode:** In NC mode, the environment status does not change rapidly. For instance, when a vehicle is following a (leading) vehicle at uniform velocity, parameters like Distance of Separation (DoS) and velocity do not change rapidly.
- **Safety-Critical (SC) Mode:** In SC mode, the environment status varies rapidly and hence tasks execute more frequently to get as accurate a view as possible about the environment.

Two-level data repository: The system consists of two levels of data store: Environment Data Repository (EDR) and Derived Data Repository (DDR). EDR is an active entity, storing the data pertaining to environment (i.e., raw data collected by the sensors). EDR contains base/raw data items and (procedures for) data derivation tasks. DDR acts as a global database for the system (and stores the information (derived data) that is derived from the data stored in EDR). The detailed system support is described in [1].

Tasks and derived data update: The system has four types of tasks: (T1) periodic tasks to read sensor values and update EDR and periodic communication receive task, (T2) sporadic on-demand update tasks for computing the derived data and updating DDR, sporadic communication transmit task and controller task for maintaining safe DoS and ensuring safe merging, (T3) periodic lower-level controller tasks for giving commands to vehicular robot and (T4) other low priority tasks which are executed only when the system is idle.

Scheduling tasks in different modes: AMC is a safety-enhancing feature and the characteristics of tasks (period, worst-case execution time) are known a priori. Hence, an algorithm that provides offline guarantee to meet task deadlines is a good choice for our system. We found the algorithm in [2, 6] as the best match for our task set—it provides offline guarantees to both periodic and sporadic tasks and schedules them online to incorporate low priority tasks if possible.

5.2 Inter-Vehicle Communication Scheme

We saw in Sect. 3 that our algorithms need a wireless communication protocol for communicating profile and other information to other vehicles (and to the infrastructure node). We have chosen Dedicated Short Range Communication (DSRC) based wireless protocol [7] for this purpose as it is being actively studied by researchers for safety-related inter-vehicle communication (e.g., PATH program at UC Berkeley [8]). We were also influenced by the fact that IEEE is adapting this protocol for

Vehicle-Vehicle (V-V) and Roadside-Vehicle (R-V) communication [9]. However, we would like to mention that our algorithms does not depend on DSRC or any particular communication protocol for that matter; any established wireless protocol can be used in place of DSRC. Since, we did not find any open DSRC standard specification; we have carried out our work based on [7]. In particular, we have used Asynchronous Fixed Repetitions with Carrier Sensing (AFR-CS) protocol [7] since it has less Probability of Reception Failure (PRF) and smaller Channel Busy Time (CBT) compared to others. The protocol randomly selects K slots among n possible slots (equal length) during the lifetime of a message and the message is repeated K times.

Now we describe the communication sequence that takes place between vehicles in a single iteration of our algorithms. Here, (i) *Initiator* refers to the vehicle that initiates communication, i.e., the vehicle that reaches the magnetic belt, (ii) *Computing Node* refers to the node which runs the algorithm using the data communicated to it. In HoL and HoL_{PE}, the *Initiator* itself is the *Computing Node*; in AFS, it is the infrastructure node.

1. *Initiator* sends *MergeInitiate* message.
2. Based on the algorithm being used, specified vehicles send their profiles to the *Computing Node*:

HoL: Head vehicle from other lane sends its profile.

AFS & HoL_{PE}: All vehicles in Z_2 whose behavior is not yet decided send their profiles.

3. The *Computing Node* uses this information to compute the profile of specified vehicles:

HoL & HoL_{PE}: Profile of the winner among the two head vehicles is computed.

AFS: Profiles of specified vehicles in Z_2 are computed.

4. These profiles are sent to the specified vehicles using AFR-CS protocol.
5. The *Computing Node* then sends a *MergeStop* message to temporarily terminate the algorithm till:

HoL & HoL_{PE}: Another vehicle whose profile is not yet computed reaches the magnetic belt.

AFS: One of the lanes is entirely filled with new vehicles (whose profiles are not yet computed).

HoL & HoL_{PE}: Above procedure is repeated till the *initiator* gets inserted into the merge sequence.

Since the algorithms use wireless communication mechanism which is inherently unreliable, it is essential to analyze how the algorithms adapt to guarantee system safety when communication fails. Failures are either:

- **Intermittent:** where the reliability of the communication medium cannot be guaranteed, i.e., few messages may fail to reach the destination or
- **Prolonged:** where the communication medium is down and hence no communication is possible—it takes time to restore the medium to normalcy.

In [4] we discuss these failures in detail and show that our algorithms continue to work without much/any modifications in case of intermittent communication failure. In the case of prolonged failures, the algorithms need to be extended a bit to ensure safety of the system. The main idea behind the proposed extensions is that, as soon as vehicles realize that there is a prolonged communication failure in the system, they switch themselves to manual mode. In our algorithms, there are two different scenarios where either: (i) all the vehicles are relying on a single *infrastructure node* to take the decision, compute their profiles and communicate the same to them (AFS or Mixed Traffic) or (ii) vehicles communicate among themselves and take the decision (HoL or HoL_{PE}). In [4] we discuss each case and show how our algorithms can be enhanced to survive failures without compromising safety.

6 Experimental Setup

Java was used to implement HoL, HoL_{PE} and AFS along with DSRC based communication protocol. The low-level design of the system was implemented on *Dexter* [10], a robotic vehicular platform. It was controlled by a PC with RTLinux3.2-pre1. The scheduling algorithm discussed in [2, 6] was implemented. The *Fire-Bird* [10] robot was used to implement the AMC algorithms in addition to the Virtual Vehicle (VV) algorithm discussed in [5]. The DSRC based AFR-CS protocol was implemented on this platform for inter-vehicle communication.

The vehicular platforms had an obstacle detection range of 30 cm, moved with a maximum speed of 50 cm/s and had a wireless radio module for communication. These platforms are used to show the proof of concept of:

- Reduction in processing capacity requirement due to our dual-mode two-level data repository approach.
- Comparison of DoS maintained by the algorithms: The experiments on robots showed that while AFS and HoL are able to maintain the desired DoS between the vehicles at the merge region, VV algorithm maintains much higher DoS thereby reducing the total throughput of the system.
- Comparison of DTTI: The experiments on robots confirmed the simulation results that AFS and HoL incur lesser DTTI as compared to VV algorithm.

We also simulated all the algorithms in Java to operate in dual state, triple-zone system along with the DSRC based AFR-CS protocol for inter-vehicle communication. All the parameter settings have been chosen carefully after carrying out an extensive set of experiments. Here, we have only explained the rationale behind setting the values of zones Z_1 , Z_2 and Z_3 and have skipped similar details for other parameters. While further experimentation is regarded for determining appropriate boundaries/values for these zones, we feel that it is another dimension which we plan to address as part of our future work. However, we have pinned the values of zones Z_1 , Z_2 and Z_3 based on one important need: ability to handle maximum traffic

density. Apart from traffic density, following factors have also influenced the lengths of zone boundaries:

- Z_3 : The proposed DSRC protocol has a limit of 1000 m as its communication range. Hence, we took a pessimistic approach and choose 400 m.
- Z_2 : We have placed a bound on min. and max. velocities of the vehicles that enter Z_3 and they all require some time to stabilize, i.e., to reach SD state (where they all follow each other with a safe DoS) and hence enter Z_2 . A simple metric, standard deviation of vehicles' velocities (when they enter Z_2) was used to capture the stability. A lower deviation implies that vehicles' velocities are very similar which makes the merging process simpler.
- Z_1 : It should be large enough so that the allowed trajectory space of vehicles is also large—larger the Z_1 radius, bigger the allowed trajectory space.

We carried out experiments by varying values for Z_1 and Z_2 and measured the traffic densities that the algorithms could handle and found the following values suitable: $Z_2 = 200$ m, and $Z_1 = 150$ m. Other parameter settings were:

1. *Environmental settings*: safe DoS was set to 5 m, radii of Z_3 , Z_2 and Z_1 from the center of merge region were set to 400, 200 and 150 m respectively.
2. *Behavior of vehicles*: initial velocity of vehicles was uniformly distributed between 17 and 20 m/s; V_{MAX} , A_{MAX} and A_{MIN} were set to 30 m/s, 4 and -4 m/s². Vehicle generation rate per lane was varied from 0.2 to 1.9 veh/s.
3. *Communication protocol*: packet size was set to 100 bytes and its lifetime to 0.02 s, the transmission rate to 20 Mbps, and the number of retransmission slots for every vehicle was varied from 1 to 20.

Note that with Z_2 being 50 m and safe DoS being 5 m, at any moment at most 20 vehicles can be in Z_2 . Our algorithms, especially AFS, was able to handle high traffic density without violating safety.

7 Evaluation Results and Observations

We describe the results obtained from our simulations as well as experiments conducted on the robotic platforms.

7.1 Simulation Results

Average DTTI and External Control Time (ECT) The first experiment was conducted to test the behavior of each of the algorithms w.r.t. average DTTI and average duration of external control (ECT) at various traffic densities or vehicle generation rates (λ) for two-lane merge scenario. We also observed the traffic density beyond

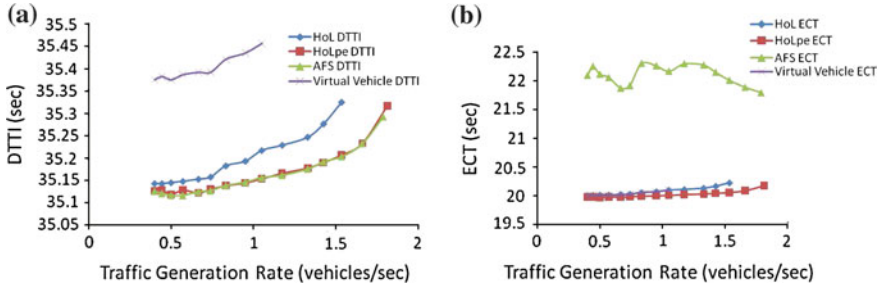


Fig. 4 DTTI and ECT for various merge algorithms. a DTTI. b ECT

which an algorithm fails to find the MS. Figure 4a, b respectively present average DTTI and ECT against λ .

A higher value of λ represents a higher traffic generation rate. The plotted graphs in both the figures depict only the *feasible regions*, i.e., they show the performance of algorithms only for those values of λ for which algorithms ensure safe merging. For example, as seen in Fig. 4a, HoL operates in the range $0 \leq \lambda \leq 1.5$ and fails to maneuver vehicles safely for higher values of λ (infeasible regions). The feasible and infeasible regions were determined after running the system long enough till it had generated around 20,000 vehicles at the specified traffic generation rate and whether it succeeded or failed in achieving the safe maneuvering for all the 20000 vehicles. As seen in the same figure, VV algorithm fails for $\lambda > 1.0$ veh/s and HoL fails to work for $\lambda > 1.5$ veh/s while the other two algorithms work even at higher densities, i.e., till $\lambda = 1.8$ veh/s. Note that VV algorithm has higher value of DTTI because it does not consider the effect of merge order on the following vehicles and it starts maintaining the safety distance right from Z_1 with the virtual vehicle. Among the other three, AFS performs slightly better with respect to DTTI. However, AFS has higher value of ECT since in AFS, on an average, vehicles switch to SD state in the middle of Z_2 , while in other algorithms they switch at the end of Z_2 .

Upper Limit on Vehicle Generation Rate (VGR) The assumption that vehicles must satisfy the safety constraint even when they enter the AoI places restriction on the maximum allowed VGR. This upper limit for VGR has been determined as follows: Let $VGR = x$ vehicles/s. Hence average time (t) between generation of two vehicles = $1/x$ s. Assuming that initial velocity of a recently generated vehicle is u and it has moved with avg acceleration a during initial time t , distance (D) traveled by this vehicle in time t is:

$$D = u \cdot t + \frac{1}{2} \cdot a \cdot t^2 = \frac{u}{x} + \frac{a}{2 \cdot x^2} \tag{6}$$

According to the safety constraint, $D > S$; using Eq. (6) and rearranging terms we get,

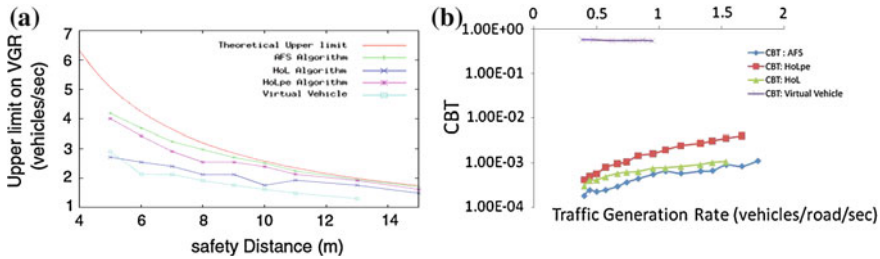


Fig. 5 Maximum VGR and the channel bandwidth usage for various algorithms. **a** Maximum VGR for various merge algorithms. **b** Communication bandwidth versus traffic density

$$S \cdot x^2 - u \cdot x - \frac{a}{2} < 0 \tag{7}$$

In order to get maximum value of x , Eq. (7) can be solved for given values of S , a and u . Solving these for $u = 25 \text{ m/s}$, $a = 4 \text{ m/s}^2$, $S = 14 \text{ m}$, we get $x \in [0, 1.86]$. In order to calculate the maximum VGR for a given safety distance that each of the algorithm can handle, several experiments were performed using the simulator and the observations are plotted in Fig. 5a. The maximum VGR that AFS, HoL_{PE}, HoL, and VV algorithms are able to handle for various values of safety distance were observed by varying the DoS from 5 to 15 m. As can be seen in Fig. 5a, the maximum traffic that can be handled for a given safety distance keeps on decreasing in the following order: AFS, HoL_{PE}, HoL, and VV. For a given safety distance, AFS is able to handle higher traffic generation rate than other algorithms.

Mixed-Traffic We evaluated the AMC system in mixed traffic scenario by varying the VM ratio and VGR. VM ratio was varied from 0 to 0.20. In Fig. 6a, DTTI is plotted for various values of VM ratio. Multiple such curves are plotted, one each at different VGR showing the feasible regions. As expected, the average DTTI value increases (albeit by a small value) with the increase in VM Ratio.

Figure 6b shows the maximum VM ratio (α) that the mixed-traffic algorithm can handle at different vehicle generation rate. Initially the maximum VM ratio that can be handled increases with the increase in VGR and later, this decreases with increase in

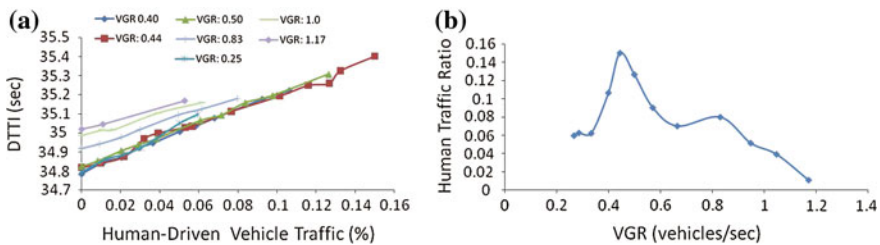


Fig. 6 Performance of the mixed traffic algorithm. **a** DTTI for mixed traffic algorithm. **b** Max. mixed traffic versus VGR

VGR. This phenomenon can be explained as follows. An encapsulation consists of AMC-controlled vehicle followed by multiple non-AMC-controlled vehicles. The non-AMC-controlled vehicle cannot be controlled directly, so the initial distance (length of an encapsulation) between the head and tail of an encapsulation is large and can be reduced only by decelerating the AMC-controlled vehicle. But there is a limitation even on this, as in the case of multiple non-AMC-controlled vehicles in an encapsulation, we will not be able to reduce separation distance between them. Thus, encapsulation length cannot be reduced beyond a certain value and if it is large, it becomes difficult for algorithm to merge vehicles (i.e., DTTI increases). This is true at low densities, so at low VGR the system works only at lower values of VM ratio. Since the initial length of encapsulation is small, system is able to handle vehicles with large VM ratio even as VGR increases. Hence, an increase in the maximum VM ratio that can be handled is observed till $VGR = 0.45$. But, as VGR increases further, flow density increases and so vehicles have lesser space for adjusting. Hence, a decrease in maximum VM ratio that can be handled by mixed-traffic algorithm is observed from this point onwards.

Communication Due to space limitations, we only present the comparison of AFS and VV algorithms in detail. Use of AFR-CS protocol requires determination of optimal value of retransmission number (K) that gives better performance w.r.t. both Channel Busy Time (CBT) and Probability of Reception Failure (PRF). In order to determine the optimal value of K , K was varied from 1 to 20 for AFS and from 1 to 10 for VV algorithm. The results for AFS and VV algorithm are presented in Fig. 7a, b respectively. On the left axis PRF is plotted, while on the right axis CBT is plotted for various values of K . The behavior of PRF and CBT has been plotted for different values of $\lambda = 1.4, 2.6, 3.6$ for AFS and $\lambda = 0.8, 1.0$ for VV algorithm. From Fig. 7a, it is clear that for $K < 6$, CBT is less but PRF is high. For $K > 6$, though PRF saturates to a constant value, CBT keeps increasing. Thus, it is evident that optimal performance w.r.t. CBT and PRF is obtained for $K = 6$. Similar pattern is also found in Fig. 7b, where $K = 4$ seems optimal. This behavior of PRF and CBT versus K is explained here:

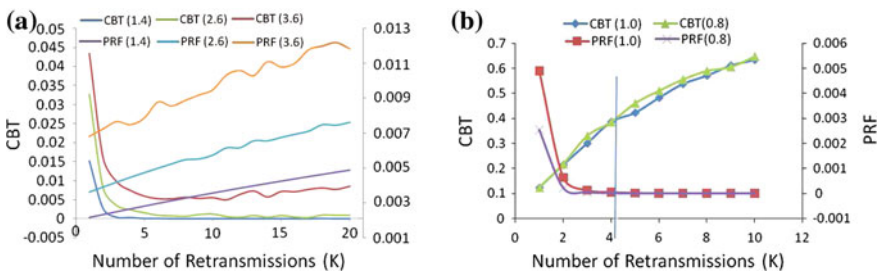


Fig. 7 Determination of retransmission number K for AFS (a) and VV (b) algorithms

- **Behavior of PRF versus K :** For low values of K , message has less number of slots, so even a small number of collisions for a message leads to message failure resulting in high PRF. As K increases, PRF decreases. Further increase in K leads to network congestion resulting in increase in the value of PRF. Thus a PRF versus K curve is U-shaped.
- **Behavior of CBT versus K :** As the number of retransmissions increases, more slots get used leading to increase in the CBT. For very high values of K , CBT versus K curve saturates to 1.

Also, as λ increases, both CBT and PRF curves shift upward—increase in λ increases message density, which in turn leads to more bandwidth usage and higher message collision (high value of PRF). These observations are in accordance with those demonstrated in [7].

In HoL algorithm, as only two vehicles communicate at any time, the message generation rate is very low and hence any value of K can be chosen. Also in HoL, changing the vehicle density does not have any impact on CBT and PRF. The channel bandwidth usage in all algorithms is shown in Fig. 5b. The bandwidth consumption (CBT) is highest in HoL_{PE}, followed by AFS and is least in HoL. This is due to the fact that in HoL_{PE}, $O(n)$ messages are sent per vehicle (where n is average number of vehicles in Z_2), while in AFS $O(1)$ messages are sent per vehicle, and in HoL at most two vehicles communicate at any moment.

7.2 Vehicular Platform Results

An experiment was done on *Dexter* [10] vehicular platform to observe the reduction in processing demand for the dual-mode two-level data repository.

The task *DistT* is an on-demand update task which derives the DoS of leading vehicle. The velocity response of a vehicle along with the mode of operation and *DistT* task invocation is depicted in Fig. 8a. We can observe from the graph that the system operates in SC mode between 0–2 s and in NC mode between 2–14 s. It again

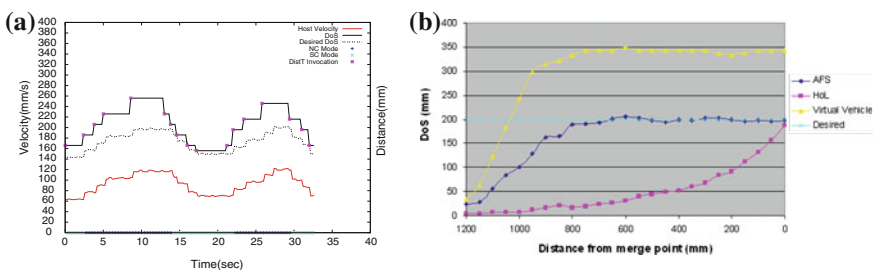
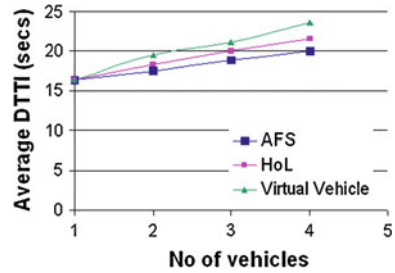


Fig. 8 Experimental results on robotic platforms. **a** Dual-mode 2-level repository results. **b** Distance of separation comparison

Fig. 9 Average DTTI comparison



enters SC mode at 14 s and continues to be in that mode till it switches back to NC mode at 22 s. Since the tasks in NC mode operate at double the periodicity compared to SC mode, there is a minimum of 50% reduction in processing requirements (as confirmed by Fig. 8a for *DistT* task) without compromising the safety.

AMC algorithms were implemented on *Firebird* [10] vehicular platform. The initial distance of the vehicles from the merge region was kept at 1200 mm. The safe/desired DoS to be maintained between two vehicles at the merge region was set to 200 mm. Figure 8b shows the comparison of DoS maintained by HoL, AFS and VV algorithm. It can be observed that HoL and AFS always maintain desired DoS when the vehicles cross the merge region, whereas VV algorithm fails to do so. AFS starts maintaining 200 mm distance even when the vehicles are at 600 mm from the merge region. We also compared average DTTI of HoL and AFS with VV algorithm by varying the number of vehicles from 1 to 4. The results are shown in Fig. 9 and it can be seen that as the number of vehicles increased from 1 to 4, AFS and HoL incurred lower DTTI as compared to VV.

To summarize, we have tried to show the feasibility of our solutions using robotic platforms. In mixed-traffic scenario when the VM ratio, α , is high, we believe that it is better to turn off AMC and control vehicles manually as it might lead to a better solution w.r.t. DTTI. However, when α is not high, our algorithms perform well as confirmed by experiments. We have performed several simulations by varying the traffic density parameter in the range of 0.4–1.8 veh/s/lane, i.e., 50–250 veh/km, which is realistic and the results show that our algorithms can handle such real-world traffic densities.

8 Related Work

The merge control application is studied in [5]. It uses the concept of virtual vehicle for ensuring safe distance criteria. But the algorithm for determining the merge order of vehicles is not provided. The work in [11] addresses the case where two lanes merge into one but again, details about algorithms and simulation parameters to compare the performance with our work are not available. In [12], a reservation based multi-agent approach is proposed for designing AMC. The drawback of this

approach is the process of repeated requests by the *driver agent* when its initial request is not met. The *intersection manager* should be smarter to make use of all the vehicles' information available to suggest an alternate space-time in the intersection instead of rejecting the request and waiting for new request from driver agent. Also, the paper does not discuss communication issues. Authors in [13] focus on risk assessment of accidents at intersections due to conflict between intention of the driver and expectation from the driver. The work exploits the sharing of information between vehicles by V2V communication. The results of the field experiments show the importance of taking interactions between vehicles into account while modeling intersection scenarios. The work in [14] focuses on priority crossing of an emergency service vehicle at an intersection. It presents a system architecture developed for cooperative vehicles applications as part of the European project, SAFESPOT and also discusses the limitations of the wireless networks, the need for clock synchronization and time stamping for such safety-related applications. Another work [15] presents an approach for collision risk estimation between vehicles by predicting the trajectories of the surrounding vehicles using the knowledge gained through communication tools. A general review of similar works in intersection safety and inter vehicular communication can be found in [16, 17]. However, none of them examine AMC problem in an end-to-end manner and hence our work is the first one to do so.

A data centric approach to the architectural design to achieve better performance for critical vehicular applications has been examined and two on-demand updating algorithms which optimistically skip unnecessary updates and hence provide increased performance are described in [3]. In particular, the work in [18] addresses the issues in the design and implementation of an active real-time database system for Electronic ECU software. Methods for the specification and run-time treatment of mode changes are discussed in [2]. We have tailored these approaches to suit our AMC application.

Real-time issues for maintaining safe DoS between vehicles and two ways to provide real-time support for efficient resource utilization, one using the dual-mode and the other using real-time data repository are presented in [19]. In this paper, we have enhanced that work by integrating both approaches to achieve better resource usage compared to both approaches and to meet deadlines of on-demand update tasks which were converted to sporadic from aperiodic.

9 Conclusions

Given the increased intelligence being built into, and the resulting increase in the number of processors in modern automobiles, there is a need to minimize the computational capacity required without affecting safety of the applications. A systematic solution to the incumbent problems is important since these by-wire applications are distributed and real-time in nature. Our work studied one such safety application, Automatic Merge Control (AMC) which ensures safe vehicle maneuver in the region where two or more roads intersect.

We proposed three merge algorithms: Head of the Lane (HoL), HoL with Propagation Effect (HoL_{PE}) and All Feasible Sequences (AFS) and also presented system support and a mechanism to determine profiles of vehicles. We also enumerated the communication requirements and showed how DSRC-based wireless communication protocol can be leveraged for the development of AMC. Another contribution is our integration of mode-change and real-time repository concepts for reducing the processing requirements. Experimental results demonstrated that HoL works only at lower traffic density whereas HoL_{PE} and AFS continue to work even at higher densities while all the algorithms gave similar performance w.r.t. DTTI. Also, experiments showed that our solutions efficiently use the computing resources without compromising real-time guarantees.

We realize that a sizable fraction of vehicles would be non-AMC-controlled in real-world and hence a system that handles mixture of both types of vehicles is essential. Considering this, we have taken a first step in this direction by providing a preliminary solution to handle such a mixed traffic where the ratio of non-AMC-controlled vehicles to AMC-controlled ones is small. Developing a full-fledged solution which can handle scenarios where this ratio is higher would be an interesting future direction. We also showed how to handle multiple-lane road scenarios and communication failures. The experiments also confirmed that our algorithms are capable of handling higher traffic densities thereby enabling efficient usage of roadways.

References

1. Shingde, V., Raravi, G., Gudhe, A., Goyal, P., Ramamritham, K.: Merge-by-wire: algorithms and system support. In: RTSS'08, pp. 25–34 (2008)
2. Fohler, G.: Flexibility in statically scheduled real-time systems. Ph.D. thesis, Technische Universitaet Wien, Austria (Apr 1994)
3. Gustafsson, T., Hansson, J.: Dynamic on-demand updating of data in real-time database systems. In: SAC'04, pp. 846–853 (2004)
4. Raravi, G., Shingde, V., Goyal, P., Ramamritham, K., Gudhe, A.: Algorithms for automatic merging of vehicles. Technical report, IIT Bombay, <http://www.cse.iitb.ac.in/krithi/papers/AMC-TR.pdf>
5. Uno, A., Sakaguchi, T., Tsugawa, S.: A merging control algorithm based on inter-vehicle communication. In: ITSC'99, pp. 783–787 (1999)
6. Isovich, D., Fohler, G.: Efficient scheduling of sporadic, aperiodic, and periodic tasks with complex constraints. In: RTSS'10, pp. 207–216 (2000)
7. Xu, Q., Mak, T., Ko, J., Sengupta, R.: Vehicle-to-vehicle safety messaging in DSRC. In: VANET'04, pp. 19–28 (2004)
8. University of California, Berkeley: Partners for advanced transportation technology. <http://www.path.berkeley.edu/> (2014)
9. Morgan, Y.: Managing DSRC and WAVE standards operations in a V2V scenario. Int. J. Veh. Technol. p. 18 (2010)
10. Nex Robotics: Robotic research platforms. <http://www.nex-robotics.com/>
11. Lebrun, J., Anda, J., Chuah, C., Zhang, M., Ghosal, D.: VGrid: vehicular AdHoc networking and computing grid for intelligent traffic control. In: VTC'05, pp. 2905–2909 (2005)
12. Dresner, K., Stone, P.: Multiagent traffic management: an improved intersection control mechanism. In: AAMAS'05, pp. 471–477 (2005)

13. Lefevre, S., Laugier, C., Ibanez-Guzman, J.: Risk assessment at road intersections: comparing intention and expectation. In: IV'12, pp. 165–171 (2012)
14. Ibanez-Guzman, J., Lefèvre, S., Mokkadem, A., Rodhaim, S.: Vehicle to vehicle communications applied to road intersection safety, field results. In: ITSC'10, pp. 192–197 (2010)
15. Lefèvre, S., Laugier, C., Ibañez-Guzmán, J.: Intention-aware risk estimation for general traffic situations, and application to intersection safety. Technical Report RR-8379, INRIA (2013)
16. Le, L., Festag, A., Baldessari, R., Zhang, W.: V2X communication and intersection safety. In: Meyer, G., Valldorf, J., Gessner, W. (eds.) *Advanced Microsystems for Automotive Applications*, pp. 97–107. Springer, Berlin (2009)
17. Salim, F.: A context-aware framework for intersection collision avoidance. Ph.D. thesis, Monash University (2008)
18. Gustafsson, T., Hansson, J.: Data management in real-time systems: a case of on-demand updates in vehicle control systems. In: RTAS'04, pp. 182–191 (2004)
19. Raravi, G., Sharma, N., Ramamritham, K., Malewar, S.: Efficient real-time support for automotive applications: a case study. In: RTCSA'06 (2006)

A Survey on Data Collection in Mobile Wireless Sensor Networks (MWSNs)

Ali Sayyed and Leandro Buss Becker

Abstract Mobile wireless sensor networks (MWSNs), a special class of WSN in which one or more component of the network is mobile, have recently grown popularity. In MWSNs, mobility plays a key role in the operation of the sensor network. As a result mobility has become an important area of research for the WSN community in recent years. Several protocols and models have been proposed in the literature which target one or other aspects of MWSNs in order to improve the process of data collection and dissemination, the ultimate goal of any sensor network. In order to develop novel and efficient techniques and protocols for mobile sensor networks we first need to have a clear understanding of the current state of the art solutions in this area. Therefore, in this chapter we present a survey on the recent advances of state of the art techniques in data collection in MWSNs.

Keywords Mobile wireless sensor networks · Mobility · Data collection · Sensor network operations

1 Introduction

In recent years Wireless Sensor Networks (WSNs) have become an established technology for a large number of real world applications, ranging from monitoring (e.g., prevention of pollution, agriculture, volcanoes, structures and buildings health), to event detection (e.g., intrusions, fire and flood emergencies) and target tracking

A. Sayyed (✉) · L.B. Becker
Department of Automation and Systems Engineering,
Federal University of Santa Catarina, Florianópolis, Brazil
e-mail: ali.sayyed@hotmail.com

L.B. Becker
e-mail: lbecker@das.ufsc.br

(e.g., surveillance and monitoring). WSNs usually consist of hundreds and in some cases thousands of battery operated tiny devices which measure and collect data from its surrounding environment and forward it to a central base station or sink.

The introduction of mobility in WSN (e.g. in [1, 2]) has attracted significant interest in recent years. Mobile nodes increase the capabilities of the WSN in many ways [2, 3]. We define mobile wireless sensor networks (MWSNs) as a special and versatile class of WSN, in which one or more than one component of the network is mobile. The mobile component can be any of the sensor nodes, relays (if any), data collectors or sink or any combination of them. From deployment to data dissemination, mobility plays a key role in almost every operation of sensor networks. For instance, a mobile node can visit other nodes in the network and collect data directly through single-hop transmissions [4]. Similarly a mobile node can move around the sensor network and collect messages from sensors, buffer them, and then transfer them to base station [5]. This significantly reduces not only collisions and message losses, but also minimizes the burden of data forwarding task by nodes and as a result spreads the energy consumption more uniformly throughout the network [6].

Allowing sensor nodes to be mobile increases the number of possible applications beyond the limits of those for which static sensors can be used. Sensors can be attached to people (for monitoring heart rate, blood pressure etc.) [7], and to animals for tracking their movements (monitoring migration patterns, feeding habits etc.) [8, 9]. Sensors may also be attached to unmanned aerial vehicles (UAVs) for surveillance or environment mapping [10].

Contributions of the book chapter. Different issues and challenges regarding MWSNs operations are already addressed in several surveys. For instance [11] cover localization issues, [12, 13] discuss routing and [14] focus on contact detection, data transfer, routing and motion control. However there are other issues which should be addressed by exploiting mobility in a sensor network. Among them are the problems of deployment, coverage, connectivity, nodes localization and cooperation. These areas are definitely very important and relevant, and are not covered in any other survey as a whole. In this chapter we try to focus on all those aspects of MWSNs operations which are affected by mobility and which directly or indirectly affect the process of data collection. These aspects includes deployment, localization, connectivity, mobile node detection, routing, mobility optimization, and cooperation among mobile and static nodes. In the rest of the chapter, in Sect. 2, we discuss some basics of MWSNs, including its classification, advantages and challenges of mobility, roles and types of mobile nodes and metrics of interests in sensor networks from applications perspective. In Sect. 3 we then present a review on different phases or aspects of MWSNs operations, along with identifying the corresponding issues and challenges. Finally, we conclude the chapter with hints to open problems.

2 MWSNs Basics

2.1 Classification of MWSNs

MWSNs can be classified in a number of ways. According to the nature of communication, MWSNs can be divided into two types [15]. The first one is called the *infrastructure network* in which mobile nodes are connected to the nearest base station within its coverage radius. The second one is called *ad-hoc network* or *infrastructure-less mobile network* in which there are no fixed routers and all mobile node(s) can move, organize and establish communication in an arbitrary manner.

MWSNs can also be classified into planer, two tiered, or three tiered network architectures [15].

- *Planer*. Heterogeneous devices, either stationary or mobile, which communicate in a multi-hop ad hoc fashion, makes a flat or planar WSN architecture. e.g. navigation system in [16].
- *Two Tiered*. This architecture consists of a set of stationary and mobile nodes. Mobile nodes, usually devices which are not limited by resources, construct an overlay network or act as data mules to help moving data and establish connectivity in the network. e.g. the NavMote system in [17].
- *Three Tiered*. In this architecture, a set of stationary sensor nodes pass data to a set of mobile devices (mobile relays), which then forward the data to a set of access points. For instance, consider a sensor network application that monitors a parking garage for parking space availability. The sensor nodes (first layer devices) broadcasts the parking space updates to mobile nodes (second layer devices e.g. smartphones and PDAs) that are in their coverage range. Finally the mobile nodes forward this data to access points (third layer devices e.g. cell towers) where the data is uploaded into a centralized database server. e.g. a three tier architecture in [5].

2.2 Advantages of Mobility in WSNs

- *Coverage and Connectivity*. In the presence of mobility, dense (re)deployment is not necessary in sensor networks. In this case mobile nodes can cope with isolated regions and cover the holes created in the connectivity of network due to dead nodes or sparseness [3].
- *Reliability*. Mobile nodes can move to different regions of the network and collect data directly through single-hop transmission, thereby reduces the number of collisions and message losses and as a result increase the probability of successful transmissions [2].
- *Lifetime*. In WSNs, nodes near to the sink deplete their energy much faster than the other nodes because they sense and forward their own data as well as data of

other nodes which are far away from sink. Mobile nodes can disperse the energy consumption and transmission more uniformly as shown in [6, 18].

- *Target Tracking*. In many real world applications of *object tracking* we need sufficient sensor nodes to be deployed along the track of the target. In addition more expensive sensing devices, e.g. camera, should be required to get more information. Nevertheless, it is infeasible to deploy large number of sensors and at the same time equip each one with a camera to tackle the situation. Controlled mobility in MWSNs can be very helpful in these type of applications as shown in [19, 20].
- *Channel Capacity*. Experiments have shown that exploiting mobility gives us greater channel capacity and data integrity due to multiple communication pathways, and less number of hops for data delivery and dissemination [21].

2.3 Challenges of Mobility in WSNs

Mobility in WSNs also introduces significant challenges, described as follows.

- *Scheduling*. Determining when an activity (detection, communication etc.) with mobile node should start/end and for what duration and with what resource is always a challenging task in MWSNs and specially when sensor nodes are sampling at different rates, in which case some nodes need to be visited more frequently than others [22].
- *Reliability*. Reliability can also be a challenge in MWSNs because the time available for detection and communication with a mobile node is scarce and short due to its movement. Paths may break frequently due to channel fading, interference and node mobility.
- *Mobility*. Mobility in MWSNs can be either uncontrollable or controllable and need to be optimized in both cases [14]. In former case, mobility patterns of mobile nodes can be learned and predicted to enhance detection and transmission process. In later case, the trajectory and speed of mobile nodes can to be optimized in order to increase network performance.
- *Localization*. Node localization is one of the most significant challenge in MWSNs [11]. Mobile nodes must continuously obtain their positions as they move in the network region.
- *Dynamic Network Topology*. Due to dynamic network topology, new routing, MAC, and scheduling protocols are needed to optimize the performance in MWSNs. For instance, static WSN routing protocols can provide the required functionality but cannot handle mobility, whereas, Mobile Ad Hoc Network (MANET) routing protocols can deal with mobility in the network but they are not designed for one way communication, which is often the case in sensor networks [12]. In addition MWSNs differ from MANET in many ways. For example in the number of nodes (density of deployment), energy requirement and traffic requirement (MWSNs are highly data driven).

2.4 Roles of Mobile Nodes in MWSNs

Mobility may exist in a sensor network in the following forms.

Mobile Sensors. Mobility may exist in ordinary or regular sensor nodes which are the sources or origin of information in WSNs. In addition these nodes may also forward or relay messages in the network. For instance, in [7–9, 23] animals/people with attached sensors, not only generate their own data, but also carry and forward data coming from other nodes which they have been previously in contact with. They eventually transfer all their data when in contact with the sink or base station.

Mobile Sinks. By Mobile Sink (or Base Station), we mean mobile nodes which are the destination or consumer of messages originated by sensors. Mobile sinks collect data sensed by sensor nodes either directly (i.e., by visiting sensors and collecting data from each of them) or indirectly (i.e., through relays or other nodes). For instance, a mobile sink is used in [4, 6] to move in the network area and collect data from sensors.

Mobile Relays. Relay nodes are neither producer nor consumer of messages in a sensor network. They perform specific task by collecting data from sensor nodes when in their coverage range, carry the data to a different location with themselves and eventually pass it to the base station. Data collection using mobile relays has been proposed in [5] where the network is based on three tiered architecture, the middle tier being represented by mobile relays.

2.5 Metrics of Interest in MWSNs

- *Network Cost.* Network cost is the first and foremost metric of interest. Adding more resources, features and complexity to network components always increase the cost of the network. On the other hand using simpler and small number of nodes, reduce network incurred expenses.
- *Lifetime.* Network/node lifetime can be defined as the time span from the deployment to the instant when the network/node is considered non-functional [24]. *Non-functional* can be defined as the instant when the first sensor dies or when some percentage of sensors dies or when loss of coverage occurs.
- *Capacity.* Network capacity is the maximum amount of information (bits) that can be transferred over a link or network path to convey data from one location in the network to another.
- *Reliability.* Reliability can be defined as the probability that the message or data will be successfully delivered to the destination without failure under stated conditions for a stated period of time.
- *Throughput.* The amount of data transferred from one place to another in a specified amount of time (the rate of successful message delivery).

- *Latency*. Latency is the amount of time it takes for a packet to travel from source to destination. It is sometimes also measured as the time required for a packet to be returned to its sender.

3 MWSNs Operations

In this section we specifically focus on all those aspects of MWSNs operations which are affected by mobility and which directly or indirectly influence the process of the data collection. These aspects include deployment, connectivity, localization, mobile node detection, routing, mobility optimization, and cooperation among mobile and static nodes

3.1 Nodes Deployment

Node deployment is the initial step in sensor network operation and mainly focus on how to best deploy the network in the sensing field. Good deployment strategies not only reduces node redundancy and network costs, but also enhance service life and data collection in the network. The aim of the deployment stage is to either cover *specific area/locations*, or to *enhance connectivity* or both.

To Maximize Connectivity. In order to work efficiently it is important for a sensor network to maintain some degree of connectivity (the ability of the sensor nodes to reach sink node). There are situations when we (re)deploy or move existing deployed sensors to change their location to better characterize the sensing area and to maximize connectivity. For instance, a system with mobile nodes targeted for improving connectivity has been proposed in [25]. In this case special mobile nodes are used to re-establish network connectivity in case of holes and faulty links.

In addition to deployment strategies used for enhancing connectivity, there are several other approaches used to predict connectivity between nodes in MWSNs. For instance, a GPS is used in [26], where the authors provide a Markov Chain to predict connectivity between mobile nodes and some fixed base stations. Similarly in [27], link quality information is used instead of GPS data to model the changes on link quality due to node mobility. On the other hand in [28] Genetic Machine Learning Algorithm is used to estimate the remaining connectivity time between neighbor nodes by combining Classifier Systems with a Markov chain model of the RF link quality. This scheme uses link quality information such as SNR, RSSI etc. and does not require any location information to perform connectivity prediction.

Area Based Coverage. Area based deployment requires a whole area of the sensing field to be covered by sensor nodes and mainly address how to deploy sensor nodes to achieve sufficient coverage of the region of interest (RoI). Area based coverage can be further subdivided into *non-uniform coverage* and *uniform coverage*.

Non-Uniform Coverage. In this case the sampling rate, data producing capability, coverage priorities etc. may be different for certain locations in the area of interest and therefore need a non-uniform coverage in case of sensor deployment.

For instance, in [29], the coverage priority of different points in the field is specified by a weighted function. Each sensor identifies coverage holes within its Voronoi polygon, and then moves in a proper direction to reduce them. As the coverage priority of different points in the field is not the same, the target location of each sensor is determined using the weighted function. Similarly in [6], a grid-quorum solution is used to quickly detect the closest redundant sensors and move it to the target location where a sensor failure or coverage hole occurs.

Uniform Coverage. In this case it is assumed that the coverage priority for different points in the field is uniform or same. Uniform deployment can be achieved using *static deployment* strategies or *dynamic deployment* strategies [30] discussed below.

In *static deployment strategy* the best location (which does not change later) for static sensor node is chosen according to the situation at hand. Static deployment is further divided in two types, one is *deterministic deployment* and other is *random deployment* [30].

In *deterministic deployment approach*, node deployment is carried out after surveyed area meshing and calculating possible node positions in advance. For instance, a grid scan approach is used in [31] where first the region of interest is divided in grids and then the best grid is selected to deploy the next sensor. Similarly a new deployment method in deterministic space with obstacles is discussed in [32]. In this case a probabilistic detection model with Watershed algorithm is used to first choose the deploying area and then using triangulation to generate the candidate positions for new sensor nodes. This results in an efficient placement of sensor nodes with coverage uniformity.

Random deployment strategies are usually used in dangerous and hazardous environment, such as forest surveillance, earthquake observation and in battlefields. In random deployment, large quantity of wireless sensor nodes are thrown (placed) in sensing area, which then form a self-organized network. For instance, random deployment based on Poisson distribution has been proposed in [33]. In this case, first of all a model of WSN node distribution is established, then a relationship between the percentage of coverage area and nodes density of the target area is determined and finally the best range of nodes density is obtained to get the optimal deployment.

In *dynamic deployment strategy*, mobile sensors after initial deployment, automatically move to optimal positions in order to improve coverage and connectivity. For instance, Cheng and Savkin in [34] discuss the coverage problem in a self-deployed MWSNs using a distributed motion coordination algorithm. In this case mobile sensors autonomously form a sensor barrier between two given landmarks to achieve the barrier coverage. Similarly in [35] two bidding protocols are designed for guiding the movement of mobile sensors in order to increase coverage to a desirable level. Static sensors identify coverage holes locally by using Voronoi diagrams and bid mobile sensors to move. Mobile sensors accept the highest bids and try to heal the largest coverage gaps.

Location Based Coverage. *Location based deployment* requires some specific locations in the sensing field to be covered by sensor nodes. In this case the deployment of sensor nodes is often performed manually. One example is the project conducted on a Hong Kong bridge [36], which is equipped with a large number of sensors (accelerometers, thermometers, strain and pressure sensors) to monitor its working conditions. In these type of applications sensors are deployed at specified locations to fulfill the required task. Since the locations of interest do not necessarily consider the network connectivity, additional relay nodes are often needed to complete the connectivity of the network and to facilitate data transmission from sensor nodes to the base station [37].

Discussion. All the previous solutions discussed in the literature assume a 2D sensing field in case of sensor deployment. While in practice and real world scenarios the sensing field is often an uneven surface with possibly a 3D structure (e.g. buildings, terrains etc.). Moreover we have usually obstacles (trees, rocks, buildings etc.) within the sensing area which can greatly degrade the transmission and hence performance of the network. These issues need to be explored deeply in order to optimally deploy sensor nodes. Similarly fault tolerance is studied individually either to enhance sensors coverage or to maximize network connectivity [38]. However in practice sensor nodes are prone to failure due to limited resources and failure of sensor nodes may lead to a premature termination of the network. Thus it is important to investigate fault tolerance jointly in order to increase network lifetime.

3.2 Nodes Localization

Accurate and low-cost sensor localization in WSN is considered important in a wide variety of applications [39]. In order to understand sensor data in a spatial context and to properly navigate mobile sensors in the sensing region, sensor position must be known. During localization, sensors nodes make some measurements and then form a map of the network. A detailed review of location estimation algorithms have been presented in [40]. Localization techniques for WSNs can be divided into *Ranged-based* and *Range-free* [41].

Ranged-based Techniques. Range-based techniques need to measure the *distance* or *angle* between each node in order to determine its geographical position. The ranging knowledge can be obtained using a number of different techniques. For instance, RSS [42], TOA [43], AOA [44] etc.

- *Received Signal Strength (RSS).* RSS is defined as the measured voltage or power (i.e., the squared magnitude of the signal strength) by a receiver circuit. RSS measurements are relatively simple and inexpensive but unpredictable and can be done by each node receiver during normal data communication without consuming additional resources [39].
- *Time of Arrival (TOA).* TOA is the time at which a signal first arrives at a receiver. It is the time of transmission plus propagation delay and is equal to the transmitter-

receiver distance divided by the propagation velocity. Receivers can accurately estimate the arrival time for line-of-sight signal, but this estimation is spoiled by additive noise and multi-path signals [39].

- *Angle of Arrival (AOA)*. AOA is the information about the direction to neighboring sensors [39]. The most common method to estimate AOA is to use an array of antennas and employ array signal processing techniques at sensor nodes. The AOA is estimated from the difference in arrival times for a transmitted signal at each of the sensor array elements. This approach requires multiple antenna elements, increasing sensor device cost and size.

Range-free Techniques. Range-free techniques use network constraints such as connectivity or anchor nodes information to estimate coordinates of the nodes instead of real ranging. For instance, in a distributed mobile sensor network discussed in [45], static sensors process all broadcasts they hear from mobile robot, including GPS data and estimate their locations using simple averaging procedure on received signal strength. Similarly another range-free based localization is proposed in [46] considering the existence of obstacles in WSNs. In this scheme, a mobile anchor node cooperates with static sensor nodes and moves actively to refine its location, while, at the same time, taking into account the relay node availability to make the best use of beacon signals. The scheme effectively enhance accuracy and minimize the effects of obstacles on node localization by using a relay node and a novel convex position estimation algorithm. A detail survey on range-free localization is presented in [47].

Localization algorithms can also be divided into *centralized* and *distributed* [39] algorithms. Centralized algorithms collect measurements at a central processor before any calculation and estimation is done while distributed algorithms require sensors to share information only with their neighbors, but possibly repeatedly. Distributed algorithms are useful in case where no central processor is available to handle the calculations or when the sensor network is large enough. Performance of localization algorithms mainly depends on the size and density of sensor network, the measurement and localization algorithms used and possibly the environment under consideration [39].

Discussion. There are several applications which rely heavily on position information of network nodes. For instance, firefighters tracking each other in a smoke-filled room, soldiers finding each other in battlefield and rescue staff locating each other in some harsh environment or natural disaster, all need accurate and reliable location information. For this purpose a combination of different methods and tools from different discipline such as communication theory, information theory, signal processing, and statistics can be used to realize accurate, reliable, and efficient network localization. Similarly cooperative localization techniques based on cooperation between static and mobile sensor nodes in MWSNs should be developed with sufficient accuracy and affordable complexity. In addition, secure localization and navigation is very important for security and military applications.

3.3 Detection of Mobile Nodes

The goal of the detection phase is to correctly and efficiently discover and identify mobile nodes as soon as they enter the communication range of static nodes. The contact time between mobile nodes and static nodes is the time for which they both are in communication range. Contact time consists of detection time (the time required for detecting the presence of a mobile node) and communication time. It is also equally important to minimize the detection time because in this way we can increase the communication time (in which actual data transfer occur between the mobile node and other node). Different techniques can be used to perform the detection phase. First, it is possible to design *general detection protocols*, which can detect mobile nodes irrespective and without knowing anything about its mobility pattern. Secondly, mobility pattern of mobile nodes can also be exploited to design *knowledge based detection protocols*.

General Detection Protocols. General detection protocols can be subdivided into *Strictly Scheduled*, *Loosely Scheduled*, and *On-Demand* [14].

Strictly Scheduled. In this case the static and mobile nodes agree on a specific time at which the data transfer may initiate. This is feasible when mobile nodes follow a very strict schedule and other nodes know exactly when mobile nodes will be in their communication range. For instance, a strictly scheduled detection protocol is implemented in [1] in which mobile nodes are assumed to be on board of public transportation shuttles that visit sensor nodes according to a tight schedule. In this way the sensor nodes could calculate the exact active time and wake up accordingly.

Strictly scheduled protocols are usually simpler to implement and are very energy efficient because they only need to exchange schedules and timetable. Such approaches require strict synchronization and accurate mobility pattern for mobile nodes to obey agreed schedules. However, this assumption is often difficult to hold in practice, unless the motion of mobile nodes is fully controllable. Due to these reasons the applicability of strictly scheduled schemes in real application scenarios is limited.

Loosely Scheduled. If mobile nodes do not follow strict schedules, sleep/wake up patterns can still be defined and nodes can still communicate without explicitly agreeing on a specific time table. For instance, a protocol based on periodic listening discussed in [2]. In this case the mobile node sends periodic activation messages (also called beacons messages), while static nodes periodically wake up and listen for advertisements from mobile node for a short time. If it does not hear any beacon message from a mobile node it can return to sleep, otherwise it can start transferring data to the mobile node [48].

Similarly in [49] mobile nodes uses multiple radios for sending advertisement or beacon messages. Here the beacon messages are replicated on multiple available channels. For example, in case of two channels, the mobile node use a high transmission power for the far detection channel, and a low transmission power for the

near detection channel. The proposed scheme provides a trade-off between detection time and energy expenditure.

On-Demand. In this case the static nodes does not periodically listen and look for the presence of mobile nodes. Instead the static nodes wake up when need arise, as a result of an activity initiated by mobile nodes. Two main approaches exists in this scenario, including *multiple radios* and *wake up messages* [14].

In [48], *multiple radio approach* has been used in which a long-range, high-power radio is used for data communication, while a low-range radio is used for waking up nodes. In this case the static node can afford to continuously listen on the low power wake up channel. As soon as the static sensor detects a wake up message, it powers up the data radio and starts communicating with the mobile node. Similarly Ansari et al. [50], uses *wake up message* from mobile nodes with enough power to trigger full activation of the static sensor node. The static sensor node use the power provided by wake up message to generate an interrupt which, in turn, enables the radio transceiver. These methods not only reduce the energy consumption of sensor nodes but also allow timely detection of mobile node. However, they have some limitations. The coverage range for both multi radio approach and radio-triggered activation is very short and they also require special hardware support. These factors limits the number of applications using on demand methods.

Knowledge Based Detection Protocols. The accuracy and efficiency of detecting mobile nodes can be further improved by exploiting some knowledge on the mobility pattern of mobile nodes [14]. In this case static sensor nodes observe the arrivals of mobile nodes and then try to learn and predict mobile nodes schedules, without agreeing on some specific time in advance. The difference between scheduled detection protocols and knowledge based protocol is that, in the latter derived and learned mobility pattern of the mobile node is used in detection phase.

In case of *deterministic mobility*, static nodes can learn and predict the arrival time of mobile nodes. For instance, in [51], the authors propose, as a first step, a learning phase where the static nodes follow a loosely scheduled scheme to check the presence of mobile nodes by performing periodic listening. Once a mobile node is detected, its arrival time is then saved and used to calculate the schedule of mobile node visits. Similarly in case of *random mobility*, sensor nodes can still learn the arrival time of mobile nodes but in this case the learning ability depends on the randomness in mobility. The more the randomness in mobility, the harder it is to learn and predict. In random mobility, probabilistic characterization of the mobility pattern can help in predicting arrival times. For instance, in [52], *mean* and *variance* of arrivals are exploited for the characterization of the mobility pattern.

In *stationary mobility* the arrivals of mobile nodes show some periodicity and their mobility patterns do not change with time. In this case the learning phase is needed only once in the start. On the other hand, in *dynamic mobility* the static nodes need continuous learning and adaption to the changing mobility patterns [14].

Discussion. Most of the detection schemes proposed in the literature are designed for static WSNs [48, 50]. Detection protocols specific for MWSNs still needs to be

designed and evaluated in the context of mobile nodes. Machine learning techniques need to be exploited in learning and predicting arrivals of mobile nodes. In some cases detection might not be very useful for each arrival of mobile nodes. For instance, if sensor nodes generates data less frequently than the frequency of mobile nodes arrival. In this situation the sensors can afford to ignore some arrivals of mobile node, while still successfully deliver its buffered data. This problem has not been fully exploited so far.

Finally, till this end researchers rely on using radios for detecting mobile nodes. Non-radio based detection mechanism (infrared sensors, microphone etc.) seems to be unexplored so far.

3.4 Routing

Routing is the process of selecting the best path(s) for transferring messages in a network from source to destination. The route of each message sent to sink is crucial in terms of consuming different network resources. Generally routing protocols for MWSNs draw inspiration from static WSN and mobile ad hoc networks (MANETs). Static WSN routing protocols provide the required functionality but cannot handle mobility. Whereas, MANET routing protocols can deal with mobility but they are not designed for one way communication, which is the case in sensor networks [12].

Routing protocols for MWSNs can be mainly classified based on their *network structure*, *state of information*, or *mobility*. Interested readers may refer to [13] for a detailed survey.

Classification based on Network Structure. Routing can be classified as *Flat routing* and *Hierarchical routing* on the basis of the network structure [13].

Flat Routing. In flat routing, also called data centric routing, all sensor nodes in the network behave in equal manner without any organization or hierarchy between them. All nodes collaboratively perform routing by sending queries, and hence collecting data from the sensors located in a region. Examples of flat routing protocols particularly modified to work with mobility are [21, 53].

In [53] a modified version of Optimized Link State Protocol (OLSR) is proposed. In this case routes are optimized by predicting the life of a link, with the help of direction of movement and position of mobile node. Similarly in [21], a modification of Directed Diffusion (DD) protocol is proposed using priority mechanisms to cope with mobility. It gives a higher priority to data coming from mobile as compared to the static node. This minimizes the unnecessary data communication between sender and intermediate nodes, in case the mobile node is actually approaching the sender.

Flat routing can be further divided into *Opportunistic Routing* and *Best Path routing* [13]. In the former, a set of candidates (next hop) for each destination are selected, each assigned a priority according to its closeness to the destination.

The highest priority node is chosen as the next hop when a message needs to be sent. In the later, the best path (based on some metrics of interest) is calculated and used to forward packets.

Hierarchical Routing. In hierarchical routing nodes in the network are organized into clusters on the basis of distance, energy, resources etc. Each cluster head manages and controls all the nodes within the cluster and is responsible for communication outside the cluster. This helps reducing the organization complexity and increases energy efficiency.

In *flat hierarchical routing* all the nodes in the sensor network have different responsibilities but have the same capabilities. For instance, M-Geocast proposed in [23], with multiple mobile sinks, where one of the mobile sinks, called a master sink, acts as data collector. All nodes send messages to the master sink by using simple geographic routing. In *cluster hierarchical routing* the sensor network is assumed to be a virtual network of interconnected clusters. The cluster head of each cluster controls, processes and forwards communication via other cluster heads to the base station or sink. For instance, a three layered architecture proposed in [54] with different responsibilities for each layer (data collection, routing task and data processing). In this case different types of sensor nodes have different capabilities performing different functions. *Zone hierarchical routing* is an extension of flat hierarchy in which the network is divided into different zones and then flat hierarchical routing is applied to each zone [55]. Here scalability is increased by performing distributive routing at each zone.

Classification Based on State of Information. Routing can be classified as *topology based routing* and *location based routing* on the basis of state of information [13].

Topology Based Routing. Topology refers to the network layout and is defined as the set of paths between nodes used explicitly or implicitly for data communication [56]. Topology based routing can be further divided into *proactive*, *reactive* and *hybrid* routing.

In *proactive routing* routes to all destinations are pre-computed and stored in a routing table and periodically updated. For instance, a mobility graph is used to encode knowledge about likely mobility patterns within the network in [57]. The mobility graph is then used to predict future relay nodes and pre-compute additional routing states in the network.

Reactive routing or on-demand routing calculates the route to a destination only when it is needed using route discovery and route maintenance process. For instance, in grid-based energy-efficient routing (GBEER) [58] when a sensor node detects an event, it generates the data announcement packet and sends it to the grid header. The header propagates the Data Announcement (DA) packet through the announcement quorum. A Grid-Quorum solution is used to effectively advertise and request the data for mobile sinks.

Hybrid routing combines the functionality of both proactive and reactive routing by restricting the scope of proactive procedure to the nodes local neighborhood or cluster. Within a cluster region, proactive routing is used while between clusters reactive routing is performed [59]. For instance, in [60], the sink periodically advertises a HELLO message which is saved with time stamp by the receiving nodes. In case of an event the nodes first try for any available sink in its cache, otherwise it initiates a route discovery process. The nodes with the available sink information reply and data transmission is started.

Location Based Routing. Location based routing exploit the geographic locations or position of nodes to route packets to its destination. In this case all nodes are assumed to be aware of their geographic locations in the network [13]. Location based routing falls into two categories as follows.

In *time based location update scheme*, each node periodically sends a location update to a central server. For instance, in [23] a time based location update routing scheme is used, in which each node maintain location information of all its neighbors. Each packet is marked with the location information of its destination. The forwarding node selects one of its neighbors that is closest to the destination through a locally optimal greedy algorithm.

In *distance based location update scheme*, each node sends a location update to a central server if the distance it moves exceeds a certain threshold. For instance, [61] exploits the location information of all sensor nodes, assigning each one a cost. Greedy forwarding is then used to route a packet to the base station by the sensor node.

Classification based on mobility. Mobility may exist in MWSNs in any component (e.g. regular sensors, relays, or sinks) depending on the scenario and application. In any case the routing protocols should support mobility accordingly in order to get optimal performance. Routing classification based on mobility fall into three categories, namely *network with mobile sensors*, *network with mobile sink*, and *network with mobile relays* [13].

Examples of WSN with mobile sensors are discussed in [7–9, 23], with mobile relays in [5], and with mobile sinks in [4, 62]. Mobility in different components of MWSNs is already discussed in Sect. 2.4.

Discussion. There are variety of solutions proposed in the literature, each one has its pros and cons. For instance, centralized solutions require more powerful mobile nodes in terms of resources, while decentralized solutions are more appropriate [51] in case we have more than one mobile node. Furthermore, most solutions depend on GPS to estimate location of the nodes which is not often available in practical and real world applications. Future work should also focus on issues like timely and energy efficient discovery of mobile nodes, mobile node scheduling, cooperation between mobile nodes and WSN etc.

3.5 Mobility Optimization

Mobility patterns followed by mobile nodes in sensor networks have significant impacts on the data collection process [14]. Mobility can be either *controllable* or *uncontrollable*. Uncontrolled mobility can be further divided into *deterministic* and *random*. In *deterministic mobility* the arrivals or contacts of mobile node is regular (usually periodic). For instance, the vehicles used for public transportation with attached sensor nodes to collect data from static sensors alongside road [1]. In *random mobility* the arrival of mobile node is not regular and follows some probability distribution, e.g. Poisson arrivals of mobile nodes in [63]. In this case the static nodes can restrict the detection process to the time, when the probability of mobile nodes arrival is high. On the other hand in *controlled mobility* the mobility can be controlled and fine tuned in term of speed and trajectory. In controlled mobility the condition for data collection is more favorable, since mobile nodes can visit static nodes at specific times, while at the same time can stop at nodes until they have collected all buffered data. However, different problems arise in this context, mainly how to schedule mobile nodes arrivals at static sensors and optimizing both the trajectory and speed of mobile nodes. In this regards, several schemes for mobile nodes have been proposed, such as in [21, 63], where approaches targeted for controlling mobility are defined.

Optimizing Trajectory. Mobile node trajectory can be either static or dynamic. A *static trajectory* refers to the path followed by mobile node which once defined does not change later with time. Different approaches are used in the literature for designing static trajectory for mobile nodes. For instance, in [64], a single mobile node is considered for data collection in circular dense WSN. The authors propose an optimized data collection protocol by first concluding that the best mobility strategy followed by mobile sink is the outer edge of the network, then considering jointly mobility and routing algorithms, and show that a better routing strategy uses a combination of round routes and short paths.

On the other hand *dynamic trajectory* refer to the path followed by mobile node which is updated and optimized on the fly according to circumstances and needs. In dynamic trajectory optimization the trajectory of the mobile node is updated as soon as an event is detected. An example of this approach is discussed in [49], where a mobile node moves along a default route. If a static node wants to be visited, it can send a visit request to the mobile node. The mobile node makes necessary changes in its trajectory by visiting the requesting node and then resumes its default route back. Another example is the iMouse [65], where static sensors inform the base station when they detect an anomaly. A mobile node equipped with cameras can then be sent by the base station to visit the location for further data collection. More examples on route optimization [45, 66, 67] are discussed in Sect. 3.6.

Optimizing Speed. Speed of mobile nodes can be optimized in two ways [14]. The first one, which is the simplest form of speed control is called *stop and communicate*. In this techniques when a mobile node enters the communication range of a static

node that has some data to send, it stops there and collects all buffered data. The duration of the stop depends on the data generation rate of the source node. Kansal et al. [21] propose a solution in which the speed of the mobile node can be controlled in a manner similar to stop and communicate.

The second way to optimize speed is called *adaptive speed control*. Adaptive speed control is discussed in [21] in which the speed of mobile node is changed according to the number of encountered nodes and the percentage of collected data with respect to buffered messages. Different group of nodes are made according to the amount of data collected, such as low, medium or high. The mobile node moves slowly in the group with a low percentage of collected data, while it moves faster when it is in communication range with the nodes with a high level of collected data.

Discussion. In most of the solutions proposed for mobile node navigation in the literature, the authors assume a linear path for mobile nodes, which is not usually the case in real world. In most of the cases, physical obstacles are not considered, which is also not very similar to the real world scenarios. Two other future research directions are experimenting *adaptive speed control* and *dynamic route update* for UAVs collecting data in large scale WSNs.

3.6 Cooperation

According to Oxford dictionary cooperation can be defined as *the willingness to assist*, or an *act of working together* for a common goal or mutual benefit. Cooperation between WSN and mobile robots has gained significant importance in recent years [68]. Mobile and static nodes can cooperate in a number of ways to increase the efficiency and performance of the network [69]. Mobile nodes, in cooperation with other static nodes in MWSNs can provide important benefits in sensor deployment, localization, route planning and navigation, connectivity repair and almost all aspects of sensor network operation.

For instance, in cooperative localization, mobile and static sensors nodes work together to make measurements, exchange information and then form a map of the network [40]. Similarly in [19, 20, 70] the problem of monitoring a large area using WSNs is considered where a set of mobile nodes cooperate with the static nodes in order to reliably detect and locate an event without any GPS or prior maps of the environment. In this case when static nodes detect a suspicious activity or event, they report it to a mobile node that can move closer to the suspected area and can confirm whether the event has occurred or not. In [19], mobile nodes decide their path based on their own information and measurements as well as information collected from the static sensors in a neighborhood around them. While in [20] the concepts of credit based approach (in which nodes are assigned credit values according to their distance from the event) and navigation force from the neighboring static nodes are used in optimizing the path. Similarly [70] propose two navigation algorithms.

The first uses the distance between the mobile node and each sensor node and the second uses the metric calculated from one-hop neighbors hop-counts. The mobile node periodically measure the distance or metric and move toward a point where these values become smaller and finally it reach the destination.

In [45, 66, 67], cooperation between an UAV and WSN have been discussed, where a dynamic route has been estimated for the flight of UAV. In [45], the sensor network employ mapping algorithms to compute adaptive, time-varying paths to events. Here a set of localized sensor nodes facilitate UAVs navigation by encoding path information and provides point-by-point navigation directions. Similarly the authors in [66], ensures that the UAV passes through some predefined zones and avoids forbidden zones. They use particle filters to predict the UAV trajectory, taking into account the UAV model, UAV kinematic and dynamic constraints of the UAV flight. The same issue is tackled in [67] by proposing a heuristic solution by decoupling the problem into four sub-problems. First of all, clusters of sensors are determined and then efficiently connected. Thirdly route inside the cluster is designed so that the information collection is maximized. Finally a path planner for the UAV to collect data is designed.

A two way cooperation between WSN and UAV in large scale network is proposed in [69]. In this case, the WSN deployed on the ground organize autonomously into clusters. As a routine operation in cluster based WSN, the role of the cluster head in each cluster is usually rotated periodically in order to conserve energy of the cluster head. In the proposed scheme the new cluster head candidate is selected according to (1) available energy (2) connectivity with other nodes and (3)for how long the candidate node is in communication range with UAV, according to its current trajectory. On the other hand, the UAV flight plan is updated according to the radio transmission coverage zones of the new cluster heads.

Similarly it is also desirable to introduce a team of UAVs to provide multiple cooperative data collection sinks [71]. First, all sensors in the network are divided into several clusters and each UAV is assigned as the data sink of one cluster. In this way data collection is performed in a parallel and faster way. In this case, the team of UAVs form a connected network, because they need to know which UAVs are functioning properly and to share information about which sensors are associated to which UAV.

Discussion. In most of the schemes proposed for mobile node navigation in the literature the authors assume the availability of GPS data, which is not usually the case in real world applications. UAVs are now a days under focus for collecting data in WSNs. In this context, navigation schemes and dynamic trajectory planning which does not depend on GPS data and which is based on the cooperation between mobile nodes and sensor network needs to be explored further.

Two other future directions are efficient cooperative localization and cooperative mobile sensor tracking.

4 Conclusions

In this chapter we have characterized different phases of Mobile Wireless Sensor Networks (MWSNs) operation. First we provided a general overview of MWSNs basics. We then defined and presented a review on different stages or aspects of MWSNs operation, that directly or indirectly affect the process of the data collection, the ultimate goal of any sensor network. The aspects presented here include deployment, connectivity, node localization, mobile node detection, routing data, mobility optimization, and cooperation between static and mobile nodes.

It must be highlighted that some issues have not been thoroughly addressed due to space constraints. Interested readers should target detailed surveys on particular topics. As a general remark, there are only a few solutions implemented in real world scenarios. Experimental evaluation and real-world deployments need to be further investigated. Moreover, complete solutions that can be applied out-of-box (immediately, with zero configuration) to specific application scenarios have not yet been proposed.

References

1. Chakrabarti, A., Sabharwal, A., Aazhang, B.: Using predictable observer mobility for power efficient design of sensor networks. In: Proceedings of the 2Nd International Conference on Information Processing in Sensor Networks, IPSN'03, pp. 129–145, Springer, Berlin, Heidelberg (2003)
2. Anastasi, G., Conti, M., Di Francesco, M.: Reliable and energy-efficient data collection in sparse sensor networks with mobile elements. *Perform. Eval.* **66**(12), 791–810 (2009)
3. Liu, B., Brass, P., Dousse, O., Nain, P., Towsley D.: Mobility improves coverage of sensor networks. In: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '05, pp. 300–308, ACM, New York, NY, USA (2005)
4. Chatzigiannakis, L., Kinalis, A., Nikolettseas, S.: Sink mobility protocols for data collection in wireless sensor networks. In: Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access, MobiWac '06, pp. 52–59, ACM, New York, NY, USA (2006)
5. Shah, R.C., Roy, S., Jain, S., Brunette, W.: Data mules: modeling a three-tier architecture for sparse sensor networks. In: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 30–41, May 2003
6. Wang, G., Cao, G., La Porta, T., Zhang, W.: Sensor relocation in mobile sensor networks. In: Proceeding of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), vol. 4, pp. 2302–2312, March 2005
7. Yan, H., Huo, H., Xu, Y., Gidlund, M.: Wireless sensor network based e-health system—implementation and experimental results. *IEEE Trans. Consum. Electron.* **56**(4), 2288–2295 (2010)
8. Ehsan, S., Bradford, K., Brugger, M., Hamdaoui, B., Kovchegov, Y., Johnson, D., Louhaichi, M.: Design and analysis of delay-tolerant sensor networks for monitoring and tracking free-roaming animals. *IEEE Trans. Wireless Commun.* **11**(3), 1220–1227 (2012)
9. Dyo, V., Ellwood, S.A., Macdonald, D.W., Markham, A., Trigoni, N., Wohlers, R., Mascolo, C., Pásztor, B., Scellato, S., Yousef, K.: Wildsensing: design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Trans. Sen. Netw.* **8**(4), 29:1–29:33 (2012)
10. White, B.A., Tsourdos, A., Ashokaraj, I., Subchan, S., Zbikowski, R.: Contaminant cloud boundary monitoring using network of uav sensors. *IEEE Sens. J.* **8**(10), 1681–1692 (2008)

11. Amundson, I., Koutsoukos, X.D.: A survey on localization for mobile wireless sensor networks. In: Proceedings of the 2nd International Conference on Mobile Entity Localization and Tracking in GPS-less Environments, MELT'09, pp. 235–254, Springer, Berlin, Heidelberg (2009)
12. Lambrou, T.P., Panayiotou, C.G.: A survey on routing techniques supporting mobility in sensor networks. In: 5th International Conference on Mobile Ad-hoc and Sensor Networks, 2009. MSN '09, Dec 2009, pp. 78–85
13. Sara, G.S., Sridharan, D.: Routing in mobile wireless sensor network: a survey. *Telecommun. Syst.* 1–29 (2013)
14. Di Francesco, M., Das, S.K., Anastasi, G.: Data collection in wireless sensor networks with mobile elements: a survey. *ACM Trans. Sen. Netw.*, **8**(1), 7:1–7:31 (2011)
15. Munir, S.A., Ren, B., Jiao, W., Wang, B., Xie D., Ma, J.: Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing. In: 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW '07, vol. 2, pp. 113–120, May 2007
16. Amundson, I., Koutsoukos, X., Sallai, J.: Mobile sensor localization and navigation using rf doppler shifts. In: Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments, MELT '08, pp. 97–102, ACM, New York, NY, USA (2008)
17. Fang, L., Antsaklis, P.J., Montestruque, L.A., McMickell, M.B., Lemmon, M., Sun, Y., Fang, H., Koutroulis, I., Haenggi, M., Xie, M., Xie, X.: Design of a wireless assisted pedestrian dead reckoning system—the navmote experience. *IEEE Trans. Instrum. Measur.* **54**(6), 2342–2358 (2005)
18. Gandham, S.R., Dawande, M., Prakash, R., Venkatesan, S.: Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In: Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE, vol. 1, pp. 377–381, Dec 2003
19. Lambrou, T.P., Panayiotou, C.G.: Collaborative event detection using mobile and stationary nodes in sensor networks. In: International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007, pp. 106–115, Nov 2007
20. Verma, A., Sawant, H., Tan, J.: Selection and navigation of mobile sensor nodes using a sensor network. *Pervasive Mob. Comput.* **2**(1), 65–84 (2006)
21. Aman K., Arun A.S., David D.J., Mani B.S., Deborah E.: Intelligent fluid infrastructure for embedded networks. In: Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services, MobiSys '04, pp. 111–124, ACM, New York, NY, USA (2004)
22. Arun, A.S., Aditya, R., Mani, B.S.: Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In: Proceedings of the 25th IEEE International Real-Time Systems Symposium, RTSS '04, pp. 296–305, IEEE Computer Society, Washington, DC, USA (2004)
23. Choi, L., Jung, J.K., Cho, B.-H., Choi, H.: M-geocast: Robust and energy-efficient geometric routing for mobile sensor networks. In: Proceedings of the 6th IFIP WG 10.2 International Workshop on Software Technologies for Embedded and Ubiquitous Systems, SEUS '08, pp. 304–316, Springer, Berlin, Heidelberg (2008)
24. Yunxia, C., Qing, Z.: On the lifetime of wireless sensor networks. *IEEE Commun. Lett.* **9**(11), 976–978 (2005)
25. Srinidhi, T., Sridhar, G., Sridhar, V.: Topology management in ad hoc mobile wireless networks. In: Proceedings of Real-Time Systems Symposium, Work-in-Progress Session (2003)
26. Nicholson, A.J., Noble, B.D.: Breadcrumbs: forecasting mobile connectivity. In: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, MobiCom '08, pp. 46–57, ACM, New York, NY, USA (2008)
27. Farkas, K., Hossmann, T., Legendre, F., Plattner, B., Das, S.K.: Link quality prediction in mesh networks. *Comput. Commun.* **31**(8), 1497–1512 (2008)
28. de Arajo Kaiser, G.M., Becker, L.B.: An evolutionary approach to improve connectivity prediction in mobile wireless sensor networks. *Procedia Comput. Sci.* **10**(0), 1100–1105 (2012) (ANT 2012 and MobiWIS 2012)

29. Mahboubi, H., Habibi, J., Aghdam, A.G., Sayrafian-Pour, K.: Cooperative self-deployment strategies in a mobile sensor network with non-uniform coverage priority. In: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, pp. 1–6, Dec 2011
30. Zhang, H., Liu, C.: A review on node deployment of wireless sensor network. *IJCSI Int. J. Comput. Sci. Issues* **9**(6) (2012)
31. Guo, X.M., Zhao, C.J., Yang, X.T. Sun, C., Li, M., Li, W., Zhou, C.: A deterministic sensor node deployment method with target coverage based on grid scan. *Chin. J. Sens. Actuators* **25**(1), 104–109 (2012)
32. Zhang, Y.-Z., Wu, C.-D., Long, C., Peng, J.: Research of node deployment strategy for wireless sensor network in deterministic space. *Control Decis.* **25**(11), 1625–1629 (2010)
33. Li, M., Ding, D., Guo, T.: Random deployment strategy of wireless sensor network node. *Comput. Eng.* **5**, 031 (2012)
34. Cheng, T.M., Savkin, A.V.: A distributed self-deployment algorithm for the coverage of mobile wireless sensor networks. *IEEE Commun. Lett.* **13**(11), 877–879 (2009)
35. Wang, G., Cao, G., Berman, P., La Porta, T.F.: Bidding protocols for deploying mobile sensors. *IEEE Trans. Mob. Comput.* **6**(5), 563–576 (2007)
36. Ko, J.M., Ni, Y.Q., Zhou, H.F., Wang, J.Y., Zhou, X.T.: Investigation concerning structural health monitoring of an instrumented cable-stayed bridge. *Struct. Infrastruct. Eng.* **5**(6), 497–513 (2009)
37. Han, X., Cao, X., Lloyd, E.L., Shen, C.-C.: Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Trans. Mob. Comput.* **9**(5), 643–656 (2010)
38. Wang, F., Liu, J.: Networked wireless sensor data collection: issues, challenges, and approaches. *IEEE Commun. Surv. Tutorials* **13**(4), 673–687 (2011)
39. Patwari, N., Ash, J.N., Kyperountas, S., Hero, A.O., Moses, R.L., Correal, N.S.: Locating the nodes: cooperative localization in wireless sensor networks. *IEEE Signal Process. Mag.* **22**(4), 54–69 (2005)
40. Bachrach, J., Taylor, C.: *Localization in Sensor Networks*, pp. 277–310. Wiley, New York (2005)
41. Guerrero, E., Alvarez, J., Rivero, L.: 3d-adal: A three-dimensional distributed range-free localization algorithm for wireless sensor networks based on unmanned aerial vehicles. In: 2010 Fifth International Conference on Digital Information Management (ICDIM), pp. 332–338, July 2010
42. Arias, J., Zuloaga, A., Lzaro, J., Andreu, J., Astarloa, A.: Malguki: an RSSI based ad hoc location algorithm. *Microprocess. Microsyst.* **28**(8), 403–409 (2004) (Resource Management in Wireless and Adhoc mobile networks)
43. Venkatraman, S., Caffery, J., You, H.-R.: A novel toa location algorithm using los range estimation for nlos environments. *IEEE Trans. Veh. Technol.* **53**(5), 1515–1524 (2004)
44. Niculescu, D., Nath, B.: Ad hoc positioning system (aps). In: Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, vol. 5, pp. 2926–2931 (2001)
45. Corke, P., Peterson, R., Rus, D.: Coordinating aerial robots and sensor networks for localization and navigation. In: Alami, R., Chatila, R., Asama, H. (eds.) *Distributed Autonomous Robotic Systems*, vol. 6, pp. 295–304. Springer, Japan (2007)
46. Chen, H., Shi, Q., Tan, R., Poor, H.V., Sezaki, K.: Mobile element assisted cooperative localization for wireless sensor networks with obstacles. *IEEE Trans. Wireless Commun.* **9**(3), 956–963 (2010)
47. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom '03*, pp. 81–95, ACM, New York, NY, USA (2003)
48. Schurgers, C., Tsiatsis, V., Srivastava, M.B.: Stem: Topology management for energy efficient sensor networks. In: *Aerospace Conference Proceedings, 2002. IEEE*, vol. 3, pp. 3-1099–3-1108 (2002)
49. Zhao, W., Ammar, M., Zegura, E.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: *Proceedings of the 5th ACM International Symposium on Mobile*

- Ad Hoc Networking and Computing, MobiHoc '04, pp. 187–198, ACM, New York, NY, USA (2004)
50. Ansari, J., Pankin, D., Mhnen, P.: Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications. *Int. J. Wireless Inf. Netw.* **16**(3), 118–130 (2009)
 51. Gao, S., Zhang, H., Das, S.K.: Efficient data collection in wireless sensor networks with path-constrained mobile sinks. *IEEE Trans. Mob. Comput.* **10**(4), 592–608 (2011)
 52. Jun, H., Ammar, M.H., Zegura, E.W.: Power management in delay tolerant networks: a framework and knowledge-based mechanisms. In: Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. pp. 418–429, Sept 2005
 53. Dantu, K., Sukhatme, G.: Connectivity vs. control: using directional and positional cues to stabilize routing in robot networks. In: Second International Conference on Robot Communication and Coordination, 2009. ROBOCOMM '09, pp. 1–6, March 2009
 54. Duan, F.F., Guo, F., Deng, M.-X., Yu, M.: Shortest path routing protocol for multi-layer mobile wireless sensor networks. In: International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09, vol. 2, pp. 106–110, April 2009
 55. Haerri, J., Bonnet, C.: On the classification of routing protocols in mobile ad-hoc networks. Technical Report EURECOM+1492, Eurecom, 08 (2004)
 56. Ramanathan, R., Rosales-Hain, R.: Topology control of multihop wireless networks using transmit power adjustment. In: Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), vol. 2, pp. 404–413 (2000)
 57. Kusy, B., Lee, H.J., Wicke, M., Milosavljevic, N., Guibas, L.: Predictive qos routing to mobile sinks in wireless sensor networks. In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09, pp. 109–120, Washington, DC, USA (2009) (IEEE Computer Society)
 58. Kweon, K., Ghim, H., Hong, J., Yoon, H.: Grid-based energy-efficient routing from multiple sources to multiple mobile sinks in wireless sensor networks. In: 4th International Symposium on Wireless Pervasive Computing, 2009. ISWPC 2009, pp. 1–5, Feb 2009
 59. Perkins, C.E.: Ad Hoc Networking, 1st edn. Addison-Wesley Professional (2008)
 60. Sharif, K., Dahlberg, T.A., Cao, L.: Anycast based lightweight routing protocol for mobile sink discovery in sensor networks. In: Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, pp. 1–2, Jan 2010
 61. Zou, L., Mi, L., Xiong, Z.: Pager-m: a novel location-based routing protocol for mobile sensor networks. In: Proceeding of Broadwise (2004)
 62. Khan, M.I., Gansterer, W., Haring, G.: Congestion avoidance and energy efficient routing protocol for wireless sensor networks with a mobile sink. *J. Netw.* **2**(6), 42–49 (2007)
 63. Somasundara, A.A., Kansal, A., Jea, D.D., Estrin, D., Srivastava, M.B.: Controllably mobile infrastructure for low energy embedded networks. *IEEE Trans. Mob. Comput.* **5**(8), 958–973 (2006)
 64. Luo, J., Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. In: Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), vol. 3, pp. 1735–1746, March 2005
 65. Tseng, Y.-C., Wang, Y.-C., Cheng, K.-Y., Hsieh, Y.-Y.: imouse: an integrated mobile surveillance and wireless sensor system. *Computer* **40**(6), 60–66 (2007)
 66. Cobano, J.A., Martinez-de Dios, J.R., Conde, R., Snchez-Matamoros, J.M., Ollero, A.: Data retrieving from heterogeneous wireless sensor network nodes using uavs. *J. Int. Rob. Syst.* **60**(1), 133–151 (2010)
 67. Sujit, P.B., Lucani, D.E., Sousa, J.B.: Joint route planning for uav and sensor network for data retrieval. In: 2013 IEEE International Systems Conference (SysCon), pp. 688–692, April 2013
 68. Michel, B., Marron, P.J., Ollero, A., Wolisz, A.: Cooperating embedded systems and wireless sensor networks, Wiley Online Library (2008)
 69. Martinez-De Dios, J.R., Lferd, K., De San Bernabé, A., Núñez, G., Torres-González, A., Ollero, A.: Cooperation between uas and wireless sensor networks for efficient data collection in large environments. *J. Intell. Robotics Syst.* **70**(1–4), 491–508 (2013)

70. Lee, W.-H., Hur, K., Eom, D.-S.: Navigation of mobile node in wireless sensor networks without localization. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008, pp. 1–7, Aug 2008
71. Wei, P., Gu, Q., Sun, D.: Wireless sensor network data collection by connected cooperative uavs. In: American Control Conference (ACC), 2013, pp. 5911–5916, June 2013