DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
ESCUELA SUPERIOR DE INGENIEROS
UNIVERSIDAD DE SEVILLA

Cooperación Descentralizada para Percepción Activa con Múltiples Robots

Decentralized Cooperation for Active Perception with Multiple Robots

por

Jesús Capitán Fernández

Ingeniero de Telecomunicación

PROPUESTA DE TESIS DOCTORAL
PARA LA OBTENCIÓN DEL TÍTULO DE

DOCTOR POR LA UNIVERSIDAD DE SEVILLA

SEVILLA, 2011

Directores

Dr.-Ing. Aníbal Ollero Baturone, Catedrático de Universidad

Dr.-Ing. Luis Merino Cabañas, Profesor Contratado Doctor de Universidad

UNIVERSIDAD DE SEVILLA

Memoria para optar al grado de Doctor por la Universidad de Sevilla

Autor:	**Jesús Capitán Fernández**
Título:	**Cooperación Descentralizada para Percepción Activa con Múltiples Robots**
Departamento:	**Departamento de Ingeniería de Sistemas y Automática**

V° B° Director:

Aníbal Ollero Baturone

V° B° Director:

Luis Merino Cabañas

El autor:

Jesús Capitán Fernández

A mi familia

A mis amigos

Agradecimientos

Sólo alguien que haya hecho una puede entender en profundidad lo que significa realizar una tesis doctoral. Muchos años de investigación, estudio, trabajo y sufrimiento se esconden detrás de este documento. Una tesis no es simplemente un largo y arduo proceso, consiste en la aportación de nuevas ideas, en buscar una línea de interés que desarrollar. Antes de llegar a ese punto y encontrar su camino, un doctorando puede enfrentarse a muchas frustraciones y periodos difíciles. Por eso, muchos nunca llegaron al final, porque un doctorando no sólo necesita una buena mente, sino perseverancia y capacidad para superar esos momentos en los que uno se siente solo. Por todo ello, ahora que he conseguido llegar al final, me gustaría agradecer aquí a todas aquellas personas que han hecho que esos momentos de soledad fueran los menos posibles en mi caso.

En primer lugar me gustaría agradecer a mis padres y a mis hermanos por su apoyo diario, sin ellos nada sería lo mismo. También a mis amigos, en especial a Fernando Borja, que me ha ayudado a revisar el inglés del documento. Y a Lidia, por ser tan comprensiva y compartirme a ratos con una tesis.

En segundo lugar, me gustaría dar las gracias a todos mis compañeros de laboratorio durante estos años: Pablo, Fran, José Manuel, Adri, Fernando, Iván, Joaquín y muchos más. Sin ellos las ideas y experimentos de esta Tesis no habrían sido posibles. Muy en especial a Luis Merino, que se ha volcado en cuerpo y alma para apoyarme y ayudarme en todo momento.

También me gustaría dar las gracias al Profesor D. Aníbal Ollero, no sólo por haber puesto los medios para realizar esta Tesis, sino por haberme convencido en su momento para entrar en el mundo de la investigación.

Sin duda alguna, mis dos estancias en el extranjero durante la Tesis han sido determinantes para alcanzar mi madurez científica. Mencionar en especial la ayuda de Sanjiv Singh y Ben Grocholsky de la Carnegie Mellon University en Pittsburgh; así como la de Pedro Lima y Matthijs Spaan del Instituto Superior Técnico en Lisboa.

Por último, agradecer al Ministerio de Educación de España, que ha financiado mi investigación durante cuatro años a través del programa de Formación de Personal Universitario.

Acknowledgements

Writing a thesis is an experience that can only be understood by those who have endured it themselves. It is the product of many years of challenging research, studies and work. It should not just be a long, bureaucratic procedure, but an endeavor of human curiosity, for it should provide with new ideas and develop interesting guidelines for future research. In their quest, a Ph.D. student must combine a bright mind with a persistent spirit to successfully overcome those difficult, lonely moments arising from the strenuous, and often frustrating, process of discovery. That is why, now that I have reached the end of such adventure, I would like to thank those who supported me through it.

First, I would like to thank my brother, sister and parents for their daily encouragement; this Thesis would not have been finished without them. I would also like to thank my friends, especially Fernando Borja, who kindly proof-read this Thesis, and Lidia, who was so understanding as to accept sharing me with my team of robots all these years.

Second, I would like to thank all my lab colleagues: Pablo, Fran, José Manuel, Adri, Fernando, Iván, Joaquín and many others. Without their help, this Thesis' ideas and experiments would have not been feasible. I also want to single out Luis Merino as he always went the extra mile to support me at every moment.

I must also thank Professor Aníbal Ollero, not only for providing the means for this Thesis, but also for having convinced me to join the international scientific community.

Undoubtedly, my two stays in foreign reseach centers during my Ph.D. were essential to the development of this Thesis. In particular, I shall acknowledge Sanjiv Singh and Ben Grocholsky from the Carnegie Mellon University (Pittsburgh, USA); as

well as Pedro Lima and Matthijs Spaan from the Instituto Superior Tecnico (Lisbon, Portugal).

Finally, this Thesis was made possible thanks to Spain's Ministry of Education, that funded it through its "Formación de Personal Universitario" academic programme.

Abstract

This Thesis focuses on cooperative active perception for robotic teams. Nowadays, there are many outdoor and indoor applications where teams of robots can be greatly helpful. In those cases, the complexity of the tasks or the variability of the places to access demand the cooperation among heterogeneous robots, which can gather all the required information thanks to their varied capabilities. Moreover, in cooperative active schemes, not only do the robots share information to estimate the state of the environment, but also take decisions to improve that perception.

The Thesis deals with decentralized approaches that can operate in real applications. These approaches provide more reliable solutions and can be scaled depending on the number of robots. Besides, communication constraints are tackled by different techniques, since they are a key issue in these scenarios.

In addition, this Thesis considers probabilistic models and representations to increase the robustness of the systems in uncertain domains. Existing models in the literature to perform active perception with multiple robots are presented. However, most of those models either are not scalable depending on the number of robots or make significant communication assumptions. The Thesis proposes approximate solutions that trade off optimality for applicability. In this sense, inter-robot communication is exploited, but the strict communication guarantees required by other methods are relaxed, since they are hard to meet in multi-robot domains. Moreover, techniques for alleviating the complexity of the models derived from realistic scenarios, and improving the scalability of the approaches are presented.

Finally, the Thesis presents extensive experimental results with teams of heterogeneous robots and sensors. Field and indoor experiments are shown in order to validate the methods in cooperative tracking.

Contents

List of Figures

Chapter 1

Introduction

This Chapter introduces the Thesis motivation, objectives and scope. Then, the Thesis outline and its main contributions are described, as well as the framework in which it has been developed. Finally, a complete survey of the different existing approaches to tackle similar problems has also been included.

1.1 Motivation

In many robotic applications, the use of teams made up of several robots can be beneficial. In those applications, the performance of complex tasks can require the cooperation among multiple robots. In rescue robotics for instance, the combination of different types of vehicles (see Figure 1.1), such as aerial and ground robots, is relevant in order to operate in hazardous places with difficult access. Besides, teams of networking robots provided with heterogeneous sensors can prove to be greatly helpful for surveillance applications, where bigger areas can be covered by several robots. Service robotic applications, where robots assist people in varied tasks, also benefit from teams with high diversity.

Even though the range of tasks that a robotic team can be aimed at is very wide, they all are usually based on the same two phases: perception and action. Basically, every robotic team is consists of sensors and actuators that enable the system to gather information from the environment and take the required actions to

Figure 1.1: Multiple heterogeneous robots that may be part of a multi-robot system. Clockwise, starting from the top left: autonomous helicopter, aerial quadrotor, two ground autonomous vehicles.

perform the tasks. The goals that the team must achieve also vary depending on the application. This Thesis focuses on *active perception approaches*, in which the robots must take decisions in order to improve the general perception of the whole system. Thus, active perception may mean performing sensory actions: for instance aiming a pan-and-tilt camera or choosing to execute an expensive vision algorithm; or influencing a robot's path planning, e.g., given two routes to get to a desired location, follow the more informative one.

In these systems where heterogeneous sensors or robots have to be integrated, a first issue is to create a common way of representing the information. The different entities of the team need to share this common interface so that the information can

be exchanged and merged between the platforms. This entails the development of a communication architecture as well as a mathematical framework for data fusion.

Regarding robotic platforms working for real applications, there are other two essential issues that should be taken into account: scalability and reliability. For practical reasons, the approach used should be scaled depending on the number of robots and offer certain robustness during its performance. In this sense, decentralized approaches are better suited, since the need for a central node is eliminated. First, communication constraints are alleviated, making the system both scalable and reliable. Second, the loss or inclusion of new members is simplified, making possible to deal with potential failures.

As it will be shown, in most of the existent architectures in the literature, either there is no decentralization or they are not demonstrated in practical applications. Actually, there are many decision-making approaches that are theoretically valid but only apply to reduced domains. Due to the lack of practical approaches, this Thesis intends to apply some theoretical frameworks to real robotic domains, focusing on decentralized methods that can work under communication constraints, and offer scalability and robustness at the same time.

Another issue to consider in real applications is the fact that robots have to operate in hazardous environments which are uncertain by nature. Reasoning about this uncertainty increases the flexibility of the system and makes the approaches more robust. Hence, the Thesis opts for a probabilistic framework that provides robustness to the whole system by modeling and reasoning about these uncertainties.

Even though the theoretical framework and methods developed in this Thesis are general, the Thesis outcomes are particularized for cooperative tracking problems. The motivation of this application is to test the Thesis results in particular robotic scenarios and prove the applicability of the system for real situations.

1.2 Objectives

The objectives set for this Thesis tries to accomplished are the following:

- To create a decentralized data fusion framework with a common representation of the information for all the robots in a robotic team. This information can be exchanged and fused among them. Approaches that model uncertainties will be considered.

- To study methods in order to deal with communication issues when fusing data in a decentralized manner. Convergence and differences with respect to a centralized version will also be analyzed for different network topologies.

- To close the loop and develop decision-making approaches to enable a cooperative team of robots to achieve certain goals. The final objective of this Thesis is to propose solutions in which the robots cooperate for active perception purposes. The decision-making approaches will use the cooperative perception layer mentioned above to obtain information about the environment.

- Scalability must be considered for the proposed solutions so that they can be applied to real robotic domains. Thus, scalability in line with the number of robots and the domain size are required.

- To analyze the existing approaches in the literature and search for decentralized methods for decision-making that deal with uncertainties. These previous works will be studied in order to develop the new approaches mentioned above that truly scale in line with the number of robots.

- The algorithms will be tested with real robotic platforms that need to perform cooperative missions. In these real applications, implementations that cope with communication failures and real-time issues will be considered. In this sense, the reliability of the techniques will also be demonstrated.

1.3 Outline and main contributions

This Thesis makes contributions to the field of cooperative active perception. Basically, it derives some theoretical results for decentralized approaches that deal with

uncertainty and apply them to real robotic teams with communication constraints. A summary of the contents of each Chapter is presented in following paragraphs.

Chapter 1 introduces the objectives and scope of the Thesis, that is placed within the framework of several research projects. A broad review of similar related works in the literature is also included.

First, algorithms and systems to obtain a shared perception of the environment in the robotic team are studied. Later, in **Chapter 2**, a probabilistic approach for decentralized data fusion is presented. In order to reach the same results that would be obtained in a centralized manner, the proposed approach maintains estimations from the past, in such a way that the current estimation can be rebuilt when delayed information arrives due to communication issues. Moreover, techniques to cope with communication delays and breakdowns are included for different network topologies. The contents of this Chapter have been published in (Capitan et al., 2009, 2011a).

In **Chapter 3**, the approach presented in the previous Chapter is tested in two real scenarios: urban robotics and rescue robotics. The algorithms developed in that Chapter are integrated into two different platforms in order to perform real experiments in cooperative tracking. The robustness and reliability of the implementations are also shown, given that the algorithms are tested in real systems and coping with real-time and communication constraints. The results have been published in (Maza et al., 2011a,b, 2010; Sanfeliu et al., 2010).

After these two Chapters, the problem of taking decisions that enable the team to improve its perception is tackled. **Chapter 4** defines the problem within a well-known mathematical framework that will be used later in the Thesis, *Partially Observable Markov Decision Processes* (POMDPs). Different probabilistic models are discussed for decision-making with robotic teams, and the related work about these models is reviewed thoroughly.

A first contribution on decision-making is presented in **Chapter 5**, which proposes a coordinated approach for controlling a team of robots in a decentralized manner. The decentralized data fusion scheme from previous Chapters is used, combined with decision-makers based on POMDPs. Even though each robot does not reason about

the others, the decentralized data fusion algorithm is able to arouse certain coordination among them. Results of this approach in coordinated tracking have been published in (Capitan et al., 2010, 2011b).

In **Chapter 6**, a cooperative approach for controlling multiple robots is presented. Again, a decentralized data fusion scheme together with POMDP controllers are used. However, in this case the decision-makers also exchange additional information in order to distribute different behaviors among the members of the team. This work resulted in experiments with real robots that have been published in (Capitan et al., 2011c) and submitted to (Capitan et al., 2011d).

Finally, **Chapter 7** summarizes the conclusions of the Thesis and proposes some guidelines for further research.

1.4 Thesis framework

An important part of the work in this Thesis has been developed within the framework of the projects **AWARE**[1] (platform for Autonomous self-deploying and operation of Wireless sensor-actuator networks cooperating with AeRial objEcts, IST-2006-33579) and **URUS**[2] (Ubiquitous Networking Robotics in Urban Sites, IST-045062), both of them funded by the European Commission.

The general goal of the **AWARE** project was the design, development and demonstration of a platform composed of heterogeneous systems able to operate cooperatively in disaster management scenarios (see Figure 1.2) without pre-existing infrastructure (or damaged infrastructure). The platform also provided self-deployment capabilities and integrated unmanned aerial vehicles, wireless sensor networks, ground fixed cameras, ground vehicles with actuation capabilities, etc.

In order to reach the main goal mentioned above, the project had the following technical objectives:

1. Develop a scalable and self-organized ground sensor network integrating cameras and wireless sensors.

[1] http://grvc.us.es/aware.
[2] http://urus.upc.es.

Figure 1.2: On the left, humanoid robots in an urban scenario of the URUS project. On the right, disaster management scenario for the application of the AWARE project.

2. Develop the architecture and middleware required for the cooperation of the heterogeneous systems including aerial vehicles, static sensor-actuator nodes, and mobile nodes carried by ground vehicles and people.

3. Develop functionalities for the platform, such as surveillance, localization, and cooperative tracking.

4. Develop new cooperative techniques for tasks requiring strong interactions between vehicles and between vehicles and the environment, such as lifting and transporting the same load by means of the cooperation of several Unmanned Aerial Vehicles (UAVs).

In particular, the objectives 1, 2 and 3 are highly related to this Thesis, since they involve the integration of information provided by different sensors in order to carry out cooperative applications with several entities.

The general goal of the **URUS** project was the development of new techniques for the cooperation between networking robots and human beings and/or their environment in urban areas (see Figure 1.2), in order to achieve tasks efficiently that would be too complex, time consuming or costly for single systems. The project was

focused on urban pedestrian areas, where there is a growing interest in reducing the number of cars and improving the quality of life.

The following technical objectives were considered in the project:

1. Develop an architecture for networking robots integrating cooperative urban robots, intelligent sensors (video cameras, acoustic sensors, etc.), intelligent devices (PDA, mobile telephones, etc.) and wireless communications.

2. Develop functionalities for the system, such as cooperative localization and navigation, cooperative environment perception, cooperative map building, human-robot interaction and multi-task negotiation.

3. Test the whole architecture in two different urban tasks: guiding and transportation of people; and surveillance.

Again, all the above objectives are connected to this Thesis, since they entail fusion of information from heterogeneous sensors and cooperative perception. In particular, this Thesis is more focused on the applications of cooperative tracking and surveillance. Actually, most results for cooperative tracking contained in this Thesis have been validated in the experiments of the projects AWARE and URUS, which provided an excellent framework.

Additionally, this work has been supported by the Cooperating Objects NETwork of Excellence **CONET**[3] (contract FP7-ICT-224053). The CONET Consortium is funded by the European Commission and aims at building a strong community in the area of cooperating objects (Marron et al., 2011) including research, public sector and industry partners from the areas of embedded systems, pervasive computing and wireless sensor networks. The project has instantiated several research clusters to accomplish the previous objectives. In particular, this Thesis contributes to the cluster of Mobility of Cooperating Objects, which tackles mobility issues among cooperative objects in dynamic environments. Indeed, most of the experiments of the Thesis involving decision-making have been carried out in an integrated testbed developed by the CONET Consortium (see Figure 1.3).

[3]https://www.cooperating-objects.eu.

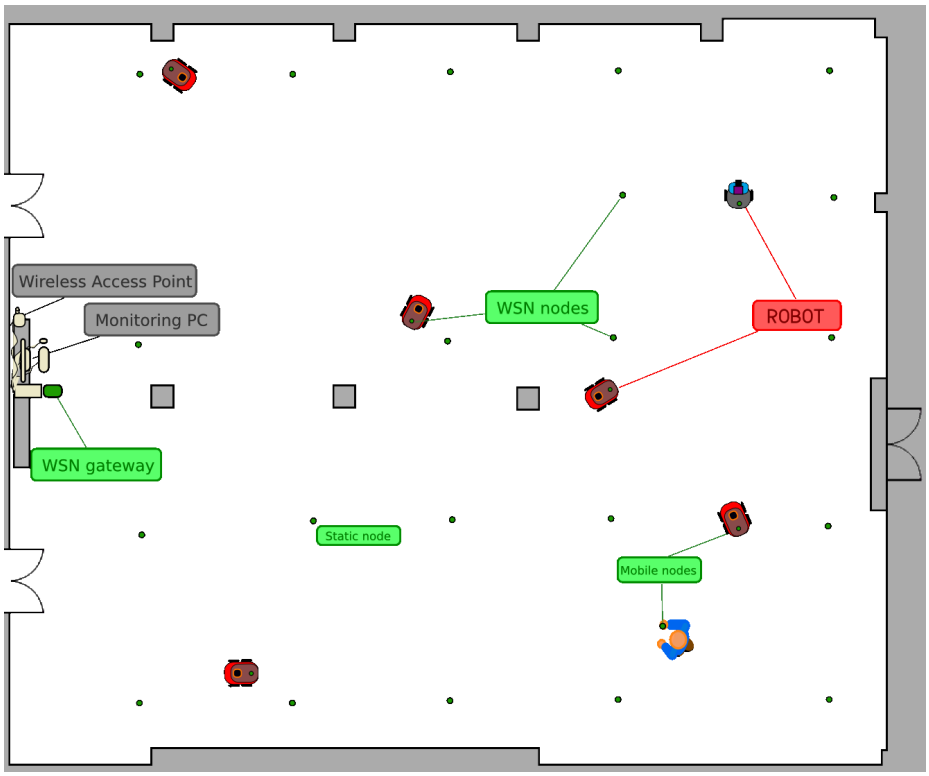


Figure 1.3: Scheme of the multi-robot testbed of CONET project. Moving robots are integrated with a Wireless Sensor Network (WSN).

This Thesis was also partially supported by the project **ROBAIR** (contract DPI2008-03847) funded by the Spanish government, and the project "**Robótica Ubícua en entornos urbanos**" (contract P09-TIC 5121), funded by the local government of Andalucía.

Finally, it should be mentioned that parts of this Thesis have been developed during two stays at the Carnegie Mellon University (June-October 2008), Pittsburgh (USA); and at the Instituto Superior Tecnico (July-October 2010), Lisbon (Portugal).

1.5 Multi-robot active perception

This Thesis focuses on cooperative approaches for active perception with multiple robots. This type of approaches can be applied to a wide variety of real problems

that require the cooperation among several robots. In order to clarify the notion of *active* perception, a general definition of the problem is defined as follows:

- There is a team of several robots that need to accomplish a certain mission. The nature of the robots can be heterogeneous, so different types of robots are considered: aerial or ground robots, moving or fixed robots, etc.
- The robots have on-board sensors that determine their perception capabilities and allow them to gather information from the environment.
- The robots can share the information they gather in order to increase their knowledge (perception). However, the available bandwidth is bounded and the robots should consider communication delays and breakdowns.
- The robots are autonomous and they have to be able to make their own decisions. These decisions should promote a cooperation among the different robots so that they can perform certain tasks together.
- In an *active* perception approach, the robot decisions are aimed at improving the perception of the whole team, i.e. they have to act actively to increase their level of information about the environment or its accuracy.

There exist many robotic applications in which active perception is a remarkable solution to achieve the given objectives. In some of them, the cooperation among multiple robots also becomes essential. For instance, in tracking or surveillance applications, getting information about a target or a certain area as accurately as possible is a typical goal. Moreover, in fire-fighting or rescue missions the quality of the information that the robots gather becomes crucial for the final success. Therefore, the different entities involved have to act to improve this critical knowledge in an efficient manner.

There are many ways of approaching this problem, and hence different methods to measure the level of information that the robots gather with each action. Therefore, depending on the particular approach or mission, the concept of improving your perception can vary. In the following section, an overview of the existing approaches is given.

1.6 Related work

There is already a wide variety of decision-making and control algorithms that tackle the problem of active perception for multi-robot and/or multi-sensor systems. This Section intends to give an overview of these previous works and classify them according to some criteria.

1.6.1 Different formulations

A first criterion to classify the approaches could be the formulation that is given to the problem. According to this, the problem could be formally described as a discounted infinite-horizon dynamic programming problem (Bertsekas, 2005) with deterministic models and solved by means of conventional optimization techniques (Furukawa et al., 2003; Spletzer and Taylor, 2003; Derenick et al., 2007) or, alternatively, it could be addressed by means of techniques incorporating other formulations.

In (Furukawa et al., 2003), a fully centralized approach in which the problem is solved by deterministic techniques is presented. It is described as a time-optimal problem, which means that the performance time is the variable to minimize. Then, an optimal controller is obtained by a control parametrization and time discretization technique in order to achieve a numerical solution.

In addition, Spletzer and Taylor (2003) present an approach to optimize the configuration of a robot team. The goal is to choose the configuration that optimize a quality function. This quality function is a measurement of the target uncertainty and its distribution is represented by particles. In order to find an optimal configuration for the quality function, gradient methods and the Simplex method are proposed. The algorithm could also be distributed but robots would need to share measurements and poses from the rest.

Another optimization framework for dynamic target tracking is considered in (Derenick et al., 2007). They propose a variable objective function that can be replaced as long as it is convex. Constraints about network connectivity and target coverage (at least one robot is tracking each target) are also tackled. The problem is

formulated as a semi-definite program so it can be solved efficiently by using generalized interior-point methods from convex optimization theory. In order to deal with the above constraints, some graphs and graph theory methods are used to guarantee connectivity.

However, for any relatively complex scenario, these deterministic techniques become intractable and alternative techniques, such as heuristic (Parker, 2002; Pirjanian and Mataric, 2000) or probabilistic ones (Stroupe and Balch, 2005; Zhao et al., 2002; Bourgault et al., 2004; Wong et al., 2005) can be used.

For instance, the formulation by Parker (2002) is heuristic and each robot calculates its own speed vector in a distributed way. The optimization problem is based on maximizing the number of observed targets during a certain time. This is applied to the tracking of multiple moving targets by a team of robots. Each member of the team can access the poses of the nearby robots and their corresponding target measurements. Then, the speed vector is calculated heuristically by weighting the other robots' and targets' poses. Thus, each robot attempts to repel the other robots at the same time that attracts the targets. Although the approach is cooperative and distributed, the teammates need to share measurements and local positions.

Pirjanian and Mataric (2000) propose another heuristic approach based on multiple objective behavior applied to multi-robot target acquisition. The problem is formulated with a task-level description and solved by fuzzy logic methods. Besides, teammates negotiate their tasks in a cooperative and distributed manner.

Techniques for decision-theoretic planning under uncertainty have also been applied increasingly to robotics in the last years (Vlassis et al., 2006). For instance, in (Stroupe and Balch, 2005), the MVERT (Most Value Estimation for Robot Teams) algorithm is presented. This is a decentralized approach in which each robot makes a decision about its next movement so that a value function is optimized. In this work, the application is mapping and tracking, and the value function is related to the target location uncertainty. Additionally, probabilistic models are used for the sensors and robots. Despite the decentralization, the members of the team need to share sensor models, targets' estimations and location data from other members.

In (Zhao et al., 2002), the application is tracking a moving target by means of a sensor network and its energy function is the one to be optimized. Due to this, the final aim is not to use the whole network all the time. Therefore, each node computes the current state belief (probabilistic formulation), decides which will be the next active node, and sends the belief to it. The decision about the next node to consider is made locally (distributed approach) by using information-utility measurements. In this sense, several information-utility measurements are proposed and compared throughout the paper.

Besides, in (Wong et al., 2005; Bourgault et al., 2004), decentralized Bayesian approaches for multi-target optimal search are described. Each robot calculates an estimation of the target position and tries to figure out which is the optimal trajectory so that a utility function is maximized. The optimal control actions are obtained by using a non-linear programming technique called Sequential Quadratic Programming (SQP). Utility functions are Bayesian as well, and they represent the probability to detect the targets. In (Bourgault et al., 2004), no control actions are shared among the robots, but their predicted measurements.

1.6.2 Centralized vs decentralized

Another relevant feature to classify works on active perception could be the distribution of the algorithm. The final goal of the system is always to solve a control/planning problem in some optimal manner. However, this control problem can be solved in a centralized or decentralized scheme. On the one hand, in a fully centralized approach (Furukawa et al., 2003), a single control problem is solved for the whole system. There is a central node that is able to access all the information. On the other hand, fully decentralized systems (Waslander et al., 2003; Stroupe and Balch, 2005; Wong et al., 2005; Bourgault et al., 2004) solve a different controller for each agent, so the algorithm is totally distributed. In the middle, semi-decentralized approaches (Burgard et al., 2005; Jung and Sukhatme, 2006) could be also considered. They exploit the distributed computational capacity that multiple platforms offer but they require a single central node to fuse information or resolve global constraints.

Waslander et al. (2003) propose a decentralized algorithm. The problem is stated as a conventional control problem and then solved by means of a penalty method. Despite being a decentralized version, each agent must know the current optimal solution for the rest of the members. In such a case, this work proves that the algorithm converges to a Nash equilibrium solution (Nash, 1950).

Burgard et al. (2005) consider a semi-decentralized algorithm. This work coordinates a team of robots to explore an unknown environment. Each robot is sent to an area so that distance and utility are optimized. Then, the utility of each area is reduced as long as other robots visit it or are close to it; and a probabilistic grid representation for the environment map is used. Furthermore, the algorithm is partially distributed and areas to explore are assigned to each robot hierarchically. It means that sub-teams of robots within communication range are created and, for each sub-team, areas are assigned by a central node.

Another example of a semi-decentralized algorithm for cooperative multi-robot tracking is (Jung and Sukhatme, 2006), where each robot calculates its own control law independently but they share position information about themselves and the targets (which may be located by some central entity). A utility function based on robots' and targets' densities is used. The density functions are evaluated along the environment, and the main goal is to choose the control law that minimize this utility function (gradient descent method). Hence, densities of robots and target are kept equalized.

In the decentralized case, further classifications can be made with regard to the information that the multiple agents share. Despite being decentralized, some algorithms require that some global information is available for all the agents; whereas in others, agents are just required to access local information. The need to maintain global sensor and robot models or any other global information can make a difference.

In (Grocholsky et al., 2003), for instance, a distinction is made between a cooperative and a coordinated solution. In the first one, a predicted optimal negotiated group decision is taken. In the second one, however, there is no negotiation or need to share global information or models. Even though decision makers act locally and do not have knowledge about the rest of robots' models or control actions, they

exchange some information that may influence implicitly other agents' subsequent decisions. Therefore, that work focuses on a coordinated information-theoretic approach to control a fleet of robots. An estimation of the environment is calculated by means of a decentralized data fusion scheme where every agent attempts to maximize locally its information gain. Within this data fusion framework, despite not being aware of the control actions carried out by the other teammates, the robots share some perception information that influences indirectly the eventual control performance of the team. Thus, the control problem can be solved locally at each agent by means of an optimal control method.

The same idea explained above about coordinated approaches is applied in (Mathews et al., 2008, 2006). They both describe the same decentralized approach for making decisions in a coordinated manner. The objective function, which is based on information gain, is optimized by a descent gradient method. Furthermore, agents do not need to know control actions carried out by the others, only local models and the impact of other agents' control actions over the objective function are considered. They assure the convergence of the algorithm under some communication assumptions.

1.6.3 Abstraction level

Decision-making algorithms can also be distinguished by the level of abstraction they consider. Thus, many of them act directly over control inputs of the robots, whereas others work in a higher level of abstraction where the commands to the robots are simply tasks. This second branch includes all the multi-robot task allocation systems, which distribute a set of tasks among a set of robots in some optimal manner. Of course, tasks are no longer basic control actions but higher-level commands, such as visiting certain points.

Marked-based techniques (Dias et al., 2006; Zlot et al., 2002) are very well-known approaches in the literature for multi-robot task allocation. The system performs a negotiation based on market rules where each task has an associated cost and a reward (which can depend on the type of robot). Finally, tasks are allocated in

such a way that a goal function is optimized. Furthermore, these techniques are quite suitable for distributed platforms since tasks and resources can be allocated and then, performed by different teammates. Despite this, a centralized entity with global knowledge about the team components and their features is usually required in order to distribute the tasks optimally. Due to this, many of these algorithms can be seen as semi-decentralized approaches.

An extensive overview and state of art of the marked-based methods for robot coordination can be found in (Dias et al., 2006). In this work, the authors go over the main advantages and drawbacks of these techniques, their varied sub-branches and some future developments.

A specific marked-based approach for cooperative exploration is detailed in (Zlot et al., 2002). Each robot proposes some goal points to visit and a negotiation is carried out by the team. In that case, travel distance and information gain are, respectively, the cost and reward considered in the negotiation stage. Even though there is a centralized entity, the algorithm is shown to be robust when communication with it is lost.

As seen above, some methods work directly over control inputs and others over a task workspace. Nevertheless, some techniques could be placed in a middle position, since they exploit features from both branches. This is the case of POMDP (*Partially Observable Markov Decision Process*) techniques, which provide an elegant way to model the interaction of an active sensor with its environment. Based on prior knowledge of the sensor models and the environment dynamics, policies which indicate the sensor how to act can be computed.

Therefore, the abstraction level of the actions which POMDPs deal with can vary. A POMDP model could be solved to determine a set of low-level control inputs over a robot, or on the contrary, the actions could be modeled simply as higher-level tasks. In fact, within the same framework, both low-level and high-level actions, can be modeled without changing the formulation. Furthermore, POMDPs allow the system to combine different kinds of objectives at the same time, just by optimizing a merged goal function. In this sense, the main goal of gathering information could be traded

off for other priorities. For instance, a fleet of robots can gather as much information as possible but also minimize the energy or the movements of the robots.

Due to this variability mentioned above, POMDPs will be chosen as a base framework for the methods in this Thesis. Thus, POMDP-based methods will be used to design active perception techniques. These techniques and their extensions for multi-robot teams will be addressed in the Chapter 4 of the Thesis.

1.6.4 Time horizon

The time horizon is another interesting consideration for decision-making algorithms. Control algorithms usually try to calculate an optimal action to take in the next time step. However, the optimization problem could consider future steps to evaluate the goal function. Therefore, when the optimization is carried out just over the current time step, the algorithm is called *greedy*, whereas *planning* is performed when actions along a time horizon are considered.

Most of the previously presented approaches apart from POMDPs perform an optimization and take a decision in a greedy manner, that is, they do not plan forward, so the time horizon is reduced to the current step. However, POMDP techniques can work with a variable time horizon and long-term goals in order to learn a control policy. This represents a higher level than just calculating the current control action, since a control policy describes the behavior of a robot along a wide variety of situations. Despite this, once the POMDP is solved and the optimal policy found, this policy translates into a set of control actions for every situation. This control actions are applied directly to the robot at each sample time, so the algorithm output works eventually over a control input level as well.

1.6.5 Summary and current works

Throughout this Section, several criteria have been used in order to differentiate among a wide range of active perception approaches. Different criteria can lead to

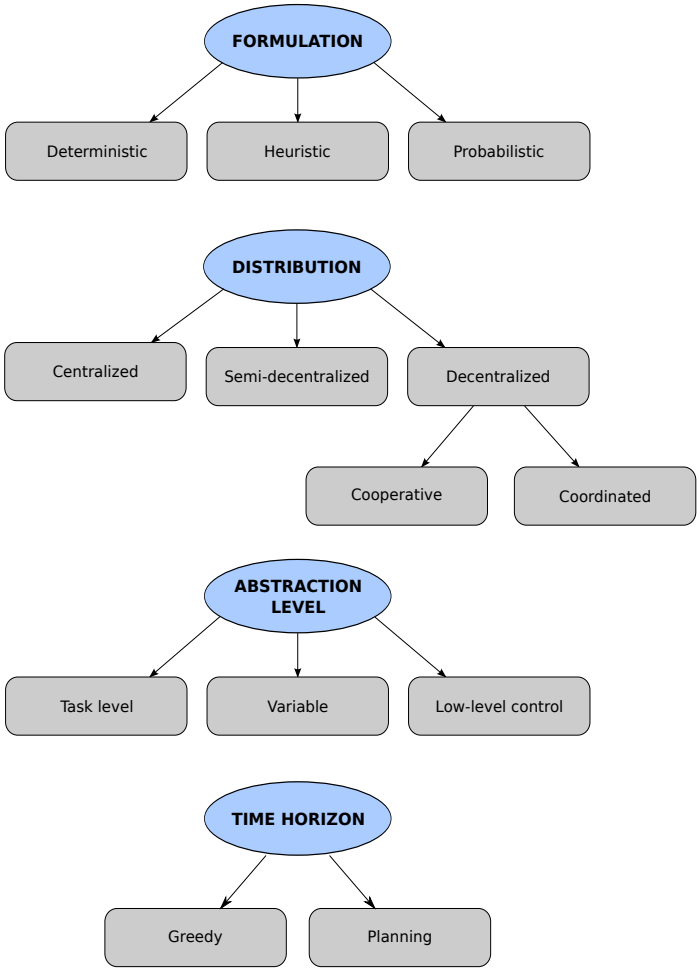


Figure 1.4: Classification criteria for active perception approaches.

different families of algorithms, so multiple classifications are possible. Figure 1.4 depicts a diagram with the algorithm features used here to establish a certain classification. The *formulation* of the algorithm indicates the nature of the mathematical framework used by each approach. Thus, conventional optimization techniques with deterministic models have been compared to heuristic and probabilistic techniques. Moreover, the analysis can also be focused on the degree of *distribution* that the algorithms present. In this sense, centralized, semi-decentralized and decentralized approaches have been cited. The *abstraction level* is highly relevant as well. Algorithms can be aimed at dealing with control actions at different levels, from low-level

Formulation	**Distribution**	**Abstraction**	**Horizon**	**References**
Deterministic	Centralized	Low-level control	Greedy	(Furukawa et al., 2003; Spletzer and Taylor, 2003; Derenick et al., 2007)
Heuristic	Semi-decentralized	Low-level control	Greedy	(Burgard et al., 2005; Jung and Sukhatme, 2006)
Heuristic	Cooperative	Low-level control	Greedy	(Parker, 2002)
Heuristic	Cooperative	Task level	Greedy	(Pirjanian and Mataric, 2000)
Probabilistic	Cooperative	Low-level control	Greedy	(Stroupe and Balch, 2005; Zhao et al., 2002; Bourgault et al., 2004; Wong et al., 2005; Makarenko et al., 2009)
Deterministic	Cooperative	Low-level control	Greedy	(Waslander et al., 2003)
Probabilistic	Coordinated	Low-level control	Planning	(Grocholsky et al., 2003; Mathews et al., 2006, 2008)
Heuristic	Semi-decentralized	Task level	Greedy	(Dias et al., 2006; Zlot et al., 2002)
Probabilistic	Variable	Variable	Planning	POMDPs

Table 1.1: Summary of previous works on active perception.

control commands to high-level task definitions. Finally, according to their *time horizon*, greedy and planning algorithms have been described. POMDP techniques have been introduce briefly within the planning group, but they will be detailed further in next Chapters.

Table 1.1 summarizes all the previous work on active perception that has been described in this Chapter. In addition to this, the features mentioned previously have been included in order to group the works into several categories.

The methods developed in this Thesis can also be placed within the above categories. A *probabilistic* formulation will be used throughout the Thesis, which only focuses on fully *decentralized* schemes. In this sense, coordinated as well as cooperative approaches are proposed. In the former, the robots do not share explicitly information about their actions, but an implicit coordination arises due to other information they may share. In the latter, robots make decisions reasoning about what the others are doing, so this kind of information is exchanged explicitly. Besides, the Thesis considers *planning* solutions that set a horizon time in the future to reason about the decisions and a *variable* abstraction level. In this sense, algorithms that reason about both low-level actions and high-level behaviors for the robots, are developed.

Finally, note that in addition to all these previous works that have been classified, additional research guidelines can be developed. For instance, in (Balch, 2000) an even higher abstraction level is considered. The behavioral diversity of a team of robots is explained as similar robots may perform some tasks in different manners. This way, control policies according to the robot perception are compared, and measurements to describe the diversity of a robot society proposed, defining as homogeneous those societies in which all the robots use similar control policies (they all behave equally). The measurements mentioned are mainly simple social and hierarchical social entropy (based on Shannon entropy). The paper explains how to calculate the diversity of a robot society when the feature that distinguishes the robots is their behavior. It also addresses the problem of quantifying the difference between robot behaviors (policies). Once it is known how to quantify the diversity of a group of robots, it could be studied how it influences the final performance of the experiments. In (Mathews et al., 2008) some of these techniques are proposed to make sub-clusters within a robot team. This is a method to cope with the exchange of information, since members of the same sub-cluster would communicate more often. Finally, the idea of taking into account similarity issues to group the different robots could be extremely profitable for future developments.

Chapter 2

Decentralized delayed-state information filter for cooperative decentralized tracking

Robotic teams can undertake active perception procedures in two basic steps: (i) the robots build a common perception of their environment by means of their sensorial capabilities; (ii) based on that information, each robot takes decisions in order to contribute to the improvement of the team perception. In this Chapter, the first problem is tackled, and a decentralized data fusion approach is developed for the robotic team to cooperatively obtain a common view of the environment from data gathered from heterogeneous sensors, which can be static or carried by robots. Particularly, a Decentralized Delayed-State Information Filter (DDSIF) is described, where full-state trajectories (that is, current and delayed states) are considered to fuse the information. This approach enables the robots to obtain an estimation equal to that provided by a centralized system and reduces the impact of communication delays and data losses into the estimation. Moreover, the filter proposed manages the information in an efficient manner, thus maintaining the communication overhead at a reasonable level.

2.1 Introduction

Scenarios in robotic applications have evolved in the last decades from very simple and controlled environments to real-world dynamic applications. The cooperation among robots and heterogeneous sensors embedded in the environment for different tasks, such as surveillance in urban scenarios (Sanfeliu and Andrade-Cetto, 2006) and disaster management (AWARE, 2006), is as a very important issue. The Thesis deals with these real scenarios involving dynamic environments and varying conditions for perception, where the robustness and reliability of autonomous perception are critical. In most cases, a single autonomous entity (e.g., a robot or a static surveillance camera) is not able to acquire all the information required for the application because of the characteristic of the particular task or the harmful conditions (e.g., loss of visibility) and, hence, the cooperation of several entities is relevant.

Therefore, the goal would be to develop a data fusion framework that combines information provided by a wide variety of heterogeneous sensors, such as cameras, laser range-finders, and other sensors. There are different approaches for this purpose. For instance, some authors employ Dempster-Shafer theory of evidence (Dempster, 1968; Shafer, 1976) for information fusion (see (Yanli et al., 2005), where the authors present a multi-robot map-building approach based on evidential reasoning). Some approaches are based on possibility theory (Zadeh, 1999), built over the arithmetic of fuzzy sets, as in the case in (LeBlanc and Saffiotti, 2009), where the authors employ such theory for cooperative localization and ball position estimation in the Robocup. Other approaches work with probabilistic beliefs but employ Consensus Theory to combine them, by using what is called an opinion pool. These approaches were largely ignored within the multi-robot research until recently (Blanco et al., 2007; Pahliani and Lima, 2007). These kinds of techniques also try to deal with the issue of disagreement (i.e., a situation in which two or more robots have inconsistent estimations).

However, most works for fusing data gathered from a network of heterogeneous sensors are based on Bayesian approaches, whereby the sensors are modeled as uncertain sources. This is the method used in this Thesis, since it makes possible to model

the dynamics of the uncertain environments that are considered. Furthermore, reasoning about uncertainties also increases the robustness and reliability of the system, which are key issues for real robotic applications.

A first potential solution would be a centralized scheme, in which each sensor just sends all its measurements to a central node where the data fusion is performed. However, this architecture presents some disadvantages that make it unsuitable for real-world applications. These drawbacks include: (i) high bandwidth requirements, especially for transmission of high-frequency motion data; (ii) limited range, since each sensor should be within communication range of the central node; and (iii) robustness issues, because a failure in the central node entails that the whole system fails.

The approach should be scalable, robust to communication failures and delays, and work properly under limited bandwidth. It has been demonstrated that decentralized approaches can cope with these requirements better than centralized ones (Makarenko et al., 2004). With these decentralized approaches, each node of the network only uses information from local sensors and shares its estimations with its neighbors without any knowledge of the topology of the whole sensor network (which will change dynamically if mobile robots are considered) or of the broadcast facilities. Thus, the need for a central node is eliminated and, since only local communications are performed, scalability is achieved. Moreover, the different members of the multi-robot platform can work more independently without the need to keep continuously communication range with a central node.

This Chapter considers the use of a delayed-state Information Filter (IF) to solve the decentralized cooperative perception problem. As it will be described, the filter considers the trajectory of the state; that is, it maintains information about past states. The main contribution is that using this full-state trajectory (not just the latest state) the nodes can recover the same information as in a centralized version, though incurring the cost of higher message sizes. The Chapter shows how using a conventional decentralized IF (without past states), the exact solution that would be obtained in a central node fusing all the measurements at their right time steps,

cannot be achieved. In the case of dynamic states, some information is proved to be missed when the fusion is carried out by using just the latest state.

This previous concept is applied efficiently to the Gaussian case thanks to the IF. The proposed delayed-state IF keeps a constant computational complexity when the trajectory grows. In addition, the sparse structure of the information matrix, whose size grows linearly with the trajectory, is exploited in order to limit the communication requirements.

Another advantage of this proposal is the possibility to cope with latency in the network, as past information can be fused. Indeed, the approach is able to deal with data that arrive out of order naturally. Besides, information about the state trajectory becomes quite important for the multi-target case, in which data association turns out to be a key issue. Since past information is maintained, this technique would enable the filter to cope with previous wrong associations. Once a wrong association is detected in a past time step, the trajectory could be recalculated from then onward.

The Chapter reviews related works and discusses general issues about decentralized fusion. Later, it describes the overall decentralized data fusion framework proposed and details the use of state trajectories in the fusion process. Finally, some simulated examples are also included to show some of the concepts explained during the Chapter.

2.2 Related work

Fusion of data gathered from a network of heterogeneous sensors is a highly relevant problem in robotics that has been widely addressed in the literature. Even though there are other possible approaches, most works are based on Bayesian approaches, where the sensors are modeled as uncertain sources. In particular, this Chapter deals with Bayesian information fusion. The simplest way to solve the problem is by fusing all the information from the network in a central entity. In (Capitan et al., 2007), for instance, a centralized Extended Kalman Filter is proposed to perform cooperative tracking. Measurements provided by cameras and a wireless sensor network are sent to a central node where the filter is running.

Nevertheless, in many works (Makarenko et al., 2004; Grocholsky et al., 2003; Grocholsky, 2002) the advantages of a decentralized scheme are highlighted. These previous works propose decentralized data fusion approaches where active sensor networks share information by means of Bayesian filters. The idea is to do it in a consistent manner, not fusing the same information twice. The so-called Channel Filters and specific network topologies are considered in all of them for this purpose. In (Grocholsky et al., 2003), the decentralized data fusion algorithm is also used to control a group of robots maximizing locally the expected information. Even though there is no explicit negotiation, the exchange of information among the members may influence others. This concept is introduced as coordinated control.

On the contrary, in (Julier and Uhlmann, 1997) the *Covariance Intersection* algorithm is presented. Basically, this consists of a conservative fusion rule that achieves consistent estimations without the need for Channel Filters or fixed network topologies. Moreover, Uhlmann (2003) presents the *Covariance Union* method, which tries to deal with disagreement in a Gaussian decentralized fusion setup.

The main issues and problems with decentralized information fusion can also be traced back to the work (Grime and Durrant-Whyte, 1994), where the Information Filter (IF, dual of the Kalman Filter) is used as the main tool for data fusion for process plant monitoring. The IF has very nice characteristics for decentralization, and for instance it has been used for decentralized mapping with aerial vehicles in (Sukkarieh et al., 2003; Nettleton et al., 2003). These works demonstrate that, for the case of static states (for instance, in mapping applications), the decentralized implementation of the IF obtains locally a final estimation that is the same as that obtained by a centralized node with access to all the information. In particular, this is applied in (Sukkarieh et al., 2003) to the project ANSER, where a team of UAVs (Unmanned Aerial Vehicles) was developed in order to perform decentralized tracking and SLAM.

In the case of dynamic states, for instance in tracking applications, it was noticed in (Rosencrantz et al., 2003; Bailey and Durrant-Whyte, 2007) that if only information about the current estimation is exchanged, information can be missed with respect to a centralized estimation. The problem is due to the fact that there is some information

not taken into account when performing the prediction steps in each fusing node. In both approaches, delayed information is considered to tackle this problem. To the best of the author's knowledge, (Bailey and Durrant-Whyte, 2007) is the closest work to the approach presented in this Chapter. It also shows how delayed information can be used to overcome some of the problems of decentralized systems. However, only results in simulation are shown. Rosencrantz et al. (2003) add a protocol that enables to selectively communicate maximally informative measurements. Hence, there is no need to send all the delayed information every time.

Furthermore, in (Nettleton, 2003) is shown how the exact centralized solution can be obtained with a single IF just when the measurements arrive in order. Thus, when a measurement arrives, some predictions are made backward to calculate the previous state and add the updated information at its right time. Once this information is incorporated, the state can be predicted forward again. However, when the information can arrive out of order, they propose to keep a history of the previous states to recover the centralized solution.

Delayed-state filters have been increasingly used by the SLAM community (Thrun et al., 2004; Eustice et al., 2006), but mainly due to the sparseness characteristics of the IF for Markov processes with high-dimensional states. In (Thrun et al., 2004), the Sparse Extended Information Filter is introduced. When the links between the robot and the features are bounded, a sparse information matrix can be maintained by this filter. Moreover, in (Eustice et al., 2006) it is demonstrated that the information matrix (for the SLAM problem) is exactly sparse in the delayed-state framework. In (Caballero et al., 2008), the authors take into account both advantages, easy decentralization and sparseness, at the same time for decentralized mapping of sensor nodes based on signal strength.

Finally, in (Makarenko et al., 2009) graphical models are proposed in order to solve similar decentralized data fusion approaches. Even though the approach is probabilistic too, the solutions are obtained by means of graphical algorithms.

2.3 Decentralized data fusion

A decentralized data fusion approach is characterized by the following (Nettleton et al., 2003):

- Each node only accesses measurements from its local sensors and obtain a local estimate.
- Each node communicates with its neighbors its local estimate, and receives estimates from its neighbors.
- There is no broadcast facility, in the sense that it cannot be ensured that the information sent will reach all the network nodes.

When a decentralized data fusion approach is considered, some relevant issues must be taken into account carefully. These issues are highly important in the sense that they could lead to inconsistent estimations. In this context, if the estimated covariances are higher than the actual ones, the estimation is considered to be consistent. In the case of inconsistent estimations, the filter may end up diverging.

First, decentralized information fusion raises the problem of rumor propagation (or double-counting of information). This problem consists of incorporating locally the same received information more than once. Actually, this would reduce the covariance of the estimation artificially, what could lead to inconsistent estimations. Therefore, when non-independent sources of information are fused, their correlation (common information) must be removed in order to assure consistent outcomes (Julier and Uhlmann, 1997).

If a pair of nodes maintain a communication link, they both will fuse all the information received from each other. However, the information previously incorporated from the other node should be discarded. For instance, a multi-path network could lead to a situation where a node receives a message which has been previously received by an alternative path (see Figure 2.1(a)).

Thus, the common information between two nodes (information previously shared by them) should be removed before fusing in order to avoid propagation, which can

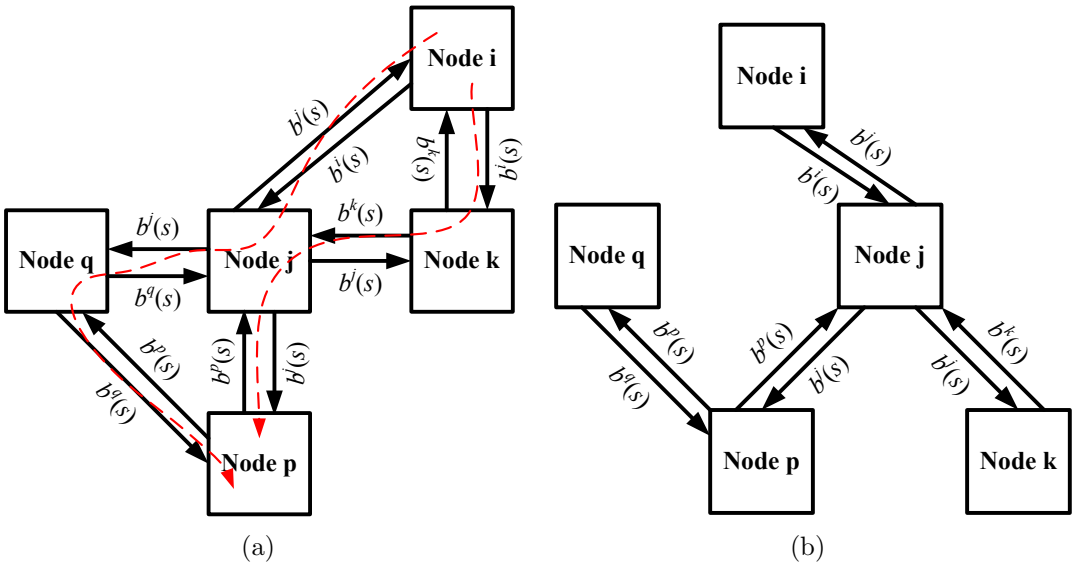


Figure 2.1: (a) Rumor propagation example due to multi-path communication. (b) Tree-like network solution.

lead to non-consistent estimations due to the loss of independence between the sources (Grime and Durrant-Whyte, 1994; Sukkarieh et al., 2003). In the literature, some typical solutions to cope with this problem can be found. The simplest one is to force a tree topology in the network, as it is depicted in Figure 2.1. In case of considering fixed network topologies such as tree-like structures, this problem can be overcome by means of Channel Filters that maintain the common information through a communication channel along time (Makarenko et al., 2004; Grocholsky et al., 2003). These fixed topologies can be too rigid for fleets of mobile robots though. Gaussian filters also provide analytical solutions for fusion under unknown common information by using the *covariance intersection* (CI) algorithm (Julier and Uhlmann, 1997), which leads to conservative estimations. Furthermore, as it will be seen, the use of delayed states allows the filter to avoid common information due to common prediction functions, which is not considered by the canonical IF.

A second important issue is the loss of information when a decentralized approach is used instead of a centralized one. In the ideal case, all the decentralized estimations of the system should converge to the centralized solution, which is considered to be optimal. This can be achieved easily when the states are static. For dynamic states

though, further constraints are required. In this case, the marginal belief of the last time step is not enough to recover the centralized solution (Rosencrantz et al., 2003), since some information can be missed during the prediction steps if measurements are not incorporated at their right moment and order. As it will be shown in detail in following Sections, a filter which considers delayed states can cope with this problem. Thus, past information which is received later due to communication delays could be added correctly into the filter.

2.4 Bayesian decentralized data fusion

In a Bayesian setup, the objective is to estimate a degree of belief $b(s_t)$ of the state s_t of the environment by using all the measurements gathered by the sensors on the N robots of a fleet[1], $z_{0:t} = [(z^1_{0:t})^T, \ldots, (z^N_{0:t})^T]^T$. This belief is the conditional probability distribution of the state given the real data, $p(s_t|z_{0:t})$. Assuming that the data gathered by the different robots at any time instant t are *conditionally independent* given the state at that instant s_t (this is a typical assumption for data fusion that requires that the state carries enough information to model the measurement process; as it will be seen, this assumption is adequate for the experiments shown in this Thesis), and the usual Markovian assumptions, the Bayes filter to compute the belief state $b(s_t)$ is given by:

$$p(s_t|z_{0:t}) = \underbrace{\eta' \prod_{i=1}^{N(t)} p(z^i_t|s_t)}_{update} \underbrace{\int p(s_t|s_{t-1}) p(s_{t-1}|z_{0:t-1}) ds_{t-1}}_{prediction} \tag{2.1}$$

with $N(t)$ the number of observations obtained at time t, and η' a normalization constant.

The belief state $b(s_{0:t})$ for the state trajectory (from time 0 up to time t) can also be derived:

[1] A sub-index indicates the time instants to which the information refers, while a super-index indicate the robot's number. Thus, $s^i_{0:t}$ would be the state up to time t for robot i, whereas s^i_t would be the state at time t.

$$p(s_{0:t}|z_{0:t}) = \eta^{''} p(s_0) \prod_{\tau=1}^{\tau=t} \Big[\prod_{i=1}^{N(\tau)} p(z_\tau^i|s_\tau) \Big] p(s_\tau|s_{\tau-1}) \tag{2.2}$$

where $p(s_0)$ is the prior and $\eta^{''}$ another normalization constant. In these centralized filters, access to all the information provided by the team at any moment is required to compute (2.1) or (2.2).

In a decentralized approach, however, each robot only uses its local data $z_{0:t}^i$ and then *shares* its belief with its neighbors. The received information from other teammates is locally fused in order to improve the local perception of the world. The belief state $b^i(s_t)$ for robot i is:

$$b^i(s_t) = p(s_t|z_{0:t}^i) = \eta_i^{'} p(z_t^i|s_t) \int p(s_t|s_{t-1}) p(s_{t-1}|z_{0:t-1}^i) ds_{t-1} \tag{2.3}$$

Considering the full trajectory it results in:

$$b^i(s_{0:t}) = \eta_i^{''} p(s_0) \prod_{\tau=1}^{\tau=t} p(z_\tau^i|s_\tau) p(s_\tau|s_{\tau-1}) \tag{2.4}$$

Comparing Equations (2.3) and (2.1), the relation between the complete belief and the local ones is given by:

$$b(s_t) = \eta^{'} \prod_{i=1}^{N} \frac{b^i(s_t)}{\int p(s_t|s_{t-1}) b^i(s_{t-1}) ds_{t-1}} \int p(s_t|s_{t-1}) b(s_{t-1}) ds_{t-1} \tag{2.5}$$

If the predicted belief is represented by $\hat{b}(s_t) = \int p(s_t|s_{t-1}) b(s_{t-1}) ds_{t-1}$, the same Equation can be written as:

$$b(s_t) = \eta^{'} \prod_{i=1}^{N} \frac{b^i(s_t)}{\hat{b}^i(s_t)} \hat{b}(s_t) \tag{2.6}$$

Figure 2.2 describes Equation (2.6) in logarithmic form. This Equation produces the same output as a centralized version only if each robot sends its belief any time they update it with new data. Otherwise, information will be missed and, clearly, the result will be different from the belief state that would be computed in a centralized

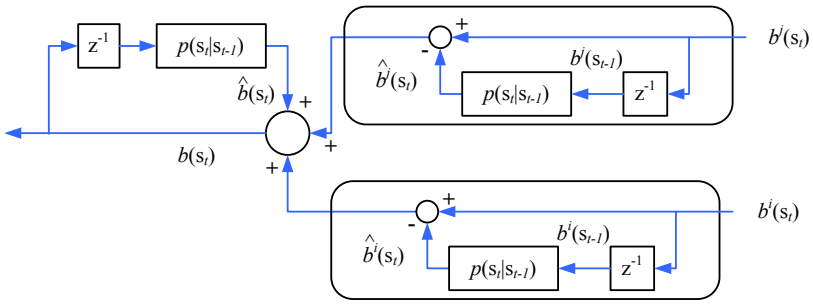


Figure 2.2: A scheme of the fusion procedure of Equation (2.5) or (2.6), in logarithmic form. The block $\mathbf{z}^{-1}$ represents a time delay. The predicted belief for each robot is subtracted from the received belief to obtain the likelihood $p(z_t^i|s_t)$, which is then added to the centralized predicted belief. In the dynamic case, delays in the transmission or missing information will lead to errors with respect to the optimal centralized estimation.

system that received all data at any time (Rosencrantz et al., 2003; Bourgault and Durrant-Whyte, 2004; Merino, 2007).

The problem is that, when the state is dynamic, the predicted belief state at any given time depends on all the past observations, so the predicted belief for a node with access to all the information is not the same as the predicted belief for each individual robot. Moreover, the importance of these differences is strongly related to the prediction model and the number of prediction steps carried out in the local nodes between consecutive communications (Bourgault and Durrant-Whyte, 2004).

As noted in (Rosencrantz et al., 2003), in a dynamic state the belief state over the full state trajectory up to time t, $b(s_{0:t})$, is required to obtain the exact solution. Therefore, comparing (2.4) and (2.2), it is possible to obtain the global belief from the local ones:

$$b(s_{0:t}) = \eta p_0(s_t) \prod_{i=1}^{N} \frac{b^i(s_{0:t})}{p_0(s_t)} \tag{2.7}$$

where $p_0(s_t) = p(s_0) \prod_{\tau=1}^{\tau=t} p(s_\tau|s_{\tau-1})$ (predicted prior). Then, if a node of the network receives all the beliefs from the other nodes, the fusion operation consists of combining all the local beliefs after removing the common information they share (the prior over

the trajectory $p_0(s_t)$). Applying this Equation, the centralized belief can be recovered exactly.

If (2.1) is considered, another possibility is to communicate to a central node only the likelihood $p(z_t^i|s_t)$ at a given instant. In this case, the problem is that the transmission of information cannot be delayed (otherwise, information is lost with respect to the fully centralized filter). Besides, with this method the use of a robot as *data mule* is lost: one robot that collects the evidence from a group of local neighbors will communicate it to other robots that could be initially disconnected from the first ones. Moreover, if the connection between two robots is lost, it will lose information that would have been available in future transmissions in the case that the robots had sent their complete beliefs.

Another advantage of using delayed states is that the belief states can be received asynchronously. Each robot can accumulate evidence ($b(s_{0:t})$), and send it whenever it is possible. However, as the state grows over time, the size of the message needed to communicate its belief also does. For the normal operation of the robots, only the state trajectory over a time interval is needed, so these belief trajectories can be bounded. However, the trajectories should be longer than the maximum expected delay in the network in order not to miss any information about past measurements.

In decentralized systems, not only does each robot receives from its neighbors, but also sends information to them. In this case, the fusion equation is slightly different. If robot i received information from j, its belief would be updated as it follows:

$$b^i(s_{0:t}) \leftarrow \eta \frac{b^i(s_{0:t}) b^j(s_{0:t})}{b^{ij}(s_{0:t})} \tag{2.8}$$

where $b^{ij}(s_{0:t})$ represents the common information between the robots (i.e., the common prior mentioned above but also information previously exchanged between the robots). This common information can be maintained by a separate filter called Channel Filter (Bourgault and Durrant-Whyte, 2004), which is basically in charge of predicting the common information up to time t. Every time a node i sends or receives information to/from another node j, its common information must be updated as follows (assuming beliefs in logarithmic form):

$$\log b^{ij}(s_{0:t}) \leftarrow \log b^{ij}(s_{0:t}) + \underbrace{\log b^{j}(s_{0:t}) - \log b^{ij}(s_{0:t})}_{j \to i} + \underbrace{\log b^{i}(s_{0:t}) - \log b^{ij}(s_{0:t})}_{i \to j} \quad (2.9)$$

where the new information received or transmitted is added to the previous common information.

The previous Channel Filter equations can only be applied if the belief network topology is tree-shaped, that is, if there is a unique path between any pair of providers and receivers (Utete and Durrant-Whyte, 1994). If there are loops in the information channels, each robot cannot determine locally if the received data were previously added. Thus, the same information could be counted twice, what can lead to over-confident estimations.

As a conclusion, all the previous equations have not, in general, an analytic solution. Next Section will present how, for Gaussian filters, there is an analytic solution, which, employing delayed states, is able, in theory, to obtain the same results as a centralized node for the case of dynamic states.

2.5 Decentralized delayed-state information filter

2.5.1 Delayed-state information filter

In the particular case of Gaussian distributions, there are analytical solutions to the previous filters, e.g. the well-known Kalman Filter (KF). The Information Filter is a more natural approach for decentralized estimation. The IF corresponds to the dual implementation of the KF. The constraints for the application of both filters are the same (Thrun et al., 2005): Markovian processes, linear prediction and measurement functions, Gaussian noises and initial Gaussian prior. Whereas the KF represents the distribution using its first μ and second Σ order moments, the IF employs the so-called *canonical representation*. The fundamental elements are the *information vector* $\xi = \Sigma^{-1}\mu$ and the *information matrix* $\Omega = \Sigma^{-1}$. Prediction and update equations for the (standard) IF can also be derived from the usual KF. In the case of

non-linear prediction or measurement, first order linearization leads to the Extended Information Filter (EIF). For more details, see (Thrun et al., 2005; Merino, 2007).

The IF presents some advantages and drawbacks when compared to the KF. One of the advantages of the canonical representation is that it can consider complete uncertainty seamlessly in the filter, by setting $\Omega_t = 0$. Furthermore, the prediction and update steps are dual in the KF and IF, in the sense that the prediction is more complicated in the IF than in the KF, but, on the other hand, the update steps are easier. Moreover, the additive nature of its update step is what makes the IF interesting for decentralized applications.

The information form also presents some interesting properties when the full-state trajectory $b(s_{0:t})$ is considered, which enable the filter to run efficiently. If the assumptions for the IF hold, it can be seen that the joint distribution over the full state is also Gaussian. The IF considering delayed states can be derived from the general Equation (2.4) (Merino, 2007). In order to consider a more general case, the EIF equations can also be used to describe the full-state trajectory filter. The following system is considered:

$$s_t = f_t(s_{t-1}) + \nu_t \tag{2.10}$$

$$z_t = g_t(s_t) + \varepsilon_t \tag{2.11}$$

where ν_t and ε_t are additive Gaussian noises. In general, f_t and g_t could be non-linear functions, so a linearization would be required. Defining the matrices A_t and M_t as $A_t = \nabla f_t(\mu_{t-1})$ and $M_t = \nabla g_t(\bar{\mu}_t)$, and knowing the information matrix and vector up to time $t-1$, $\Omega_{0:t-1}$ and $\xi_{0:t-1}$, the prediction steps are:

Algorithm 1 $(\xi_{0:t}, \Omega_{0:t}) \leftarrow \text{EIF}(\xi_{0:t-1}, \Omega_{0:t-1}, z_t)$

1: $\bar{\Omega}_{0:t} = \texttt{Add\_Block\_Matrix}(\ \Omega_{0:t-1}) + \begin{pmatrix} \begin{pmatrix} I \\ -A_t^T \end{pmatrix} R_t^{-1} \begin{pmatrix} I & -A_t \end{pmatrix} & 0 \\ 0 & 0 \end{pmatrix}$

2: $\bar{\xi}_{0:t} = \texttt{Add\_Block\_Vector}(\xi_{0:t-1}) + \begin{pmatrix} R_t^{-1}(f_t(\mu_{t-1}) - A_t\mu_{t-1}) \\ -A_t^T R_t^{-1}(f_t(\mu_{t-1}) - A_t\mu_{t-1}) \\ 0 \end{pmatrix}$

3: $\Omega_{0:t} = \bar{\Omega}_{0:t} + \begin{pmatrix} M_t^T S_t^{-1} M_t & 0 \\ 0 & 0 \end{pmatrix}$

4: $\xi_{0:t} = \bar{\xi}_t + \begin{pmatrix} M_t^T S_t^{-1}(z_t - g_t(\bar{\mu}_t) + M_t\bar{\mu}_t) \\ 0 \end{pmatrix}$

$$\bar{\Omega}_{0:t} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Omega_{t-1t-1} & \dots \\ 0 & \vdots & \ddots \end{pmatrix} + \\ + \begin{pmatrix} R_t^{-1} & -R_t^{-1}A_t & 0 \\ -A_t^T R_t^{-1} & A_t^T R_t^{-1} A_t & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.12}$$

$$\bar{\xi}_{0:t} = \begin{pmatrix} 0 \\ \xi_{t-1} \\ \xi_{0:t-2} \end{pmatrix} + \begin{pmatrix} R_t^{-1}(f_t(\mu_{t-1}) - A_t\mu_{t-1}) \\ -A_t^T R_t^{-1}(f_t(\mu_{t-1}) - A_t\mu_{t-1}) \\ 0 \end{pmatrix} \tag{2.13}$$

And, if one measurement is received, the update Equations are:

$$\Omega_{0:t} = \bar{\Omega}_{0:t} + \begin{pmatrix} M_t^T S_t^{-1} M_t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.14}$$

$$\xi_{0:t} = \bar{\xi}_{0:t} + \begin{pmatrix} M_t^T S_t^{-1}(z_t - g_t(\bar{\mu}_t) + M_t\bar{\mu}_t) \\ 0 \end{pmatrix} \tag{2.15}$$

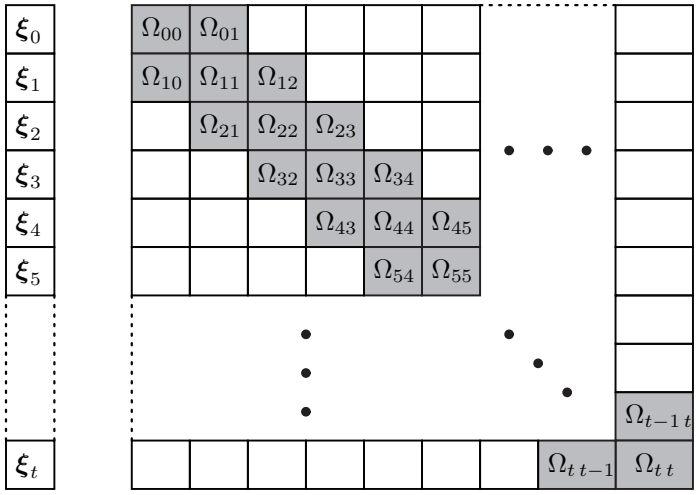


Figure 2.3: Structure of the information matrix for the full trajectory. The information matrix is a block-tridiagonal symmetric matrix, due to the Markov structure of the process.

where R_t, S_t are the corresponding covariances of the additive noises for the prediction and measurement models (2.10) and (2.11), respectively. Further details can be seen in (Merino, 2007). Then, the delayed-state EIF is summarized in Algorithm 1, where `Add_Block_Matrix` adds a block row and a block column to the previous information matrix and `Add_Block_Vector` adds a block row to the previous information vector. Note that the information matrix and vector for the full trajectory are composed of blocks (sub-matrices) of the same size as the state for a single time step. One new block is added to the information vector of the trajectory at each time step (ξ_t), whereas three are added to the information matrix ($\Omega_{tt}, \Omega_{tt-1}, \Omega_{t-1t}$).

Evidently, the state grows along time. In the general case of an information matrix, for a L-dimensional state, the storage required is $O(L^2)$. However, in this case, as it can be seen from the prediction and update equations, the matrix structure is block-tridiagonal and symmetric (see Figure 2.3) at any time, and thus the storage required is $O(L)$ (where L is the number of time steps). In general, if the measurements only depend on part of the state, the matrix will be sparse. Also, the computational complexity of the algorithm itself is $O(1)$, as the prediction and updatr computations at each time instant only involve the previous block. These considerations enable the

proposed approach to cope with the delayed states more efficiently than the classical KF.

State reduction

In certain situations, the length of the estimated trajectory should be limited, for instance due to storage or bandwidth restrictions. Therefore, a method to reduce the state whenever the size of the trajectory grows over a given threshold is required.

In order to do this, the removed part of the trajectory should be marginalized out. The marginal of a multivariate Gaussian in the canonical form can be computed in closed form (Eustice et al., 2006). Moreover, due to the structure of the information matrix for this case, the computations required only involve local block matrix operations (see Figure 2.4). In addition, this marginalization operation maintains the block-tridiagonal structure of the matrix. In general, if the information at time t is eliminated, the only blocks affected are those linked to it (that is, $t-1$ and $t+1$), following:

$$\begin{aligned}
\Omega_{t-1t-1} &\leftarrow \Omega_{t-1t-1} - \Omega_{tt-1}^T \Omega_{tt}^{-1} \Omega_{tt-1} \\
\xi_{t-1} &\leftarrow \xi_{t-1} - \Omega_{tt-1}^T \Omega_{tt}^{-1} \xi_t \\
\Omega_{t+1t+1} &\leftarrow \Omega_{t+1t+1} - \Omega_{t+1t} \Omega_{tt}^{-1} \Omega_{t+1t}^T \\
\xi_{t+1} &= \xi_{t+1} - \Omega_{t+1t}^T \Omega_{tt}^{-1} \xi_t \\
\Omega_{t+1t-1} &\leftarrow -\Omega_{t+1t} \Omega_{tt}^{-1} \Omega_{tt-1}
\end{aligned} \tag{2.16}$$

2.5.2 Decentralized information filter

The proposed EIF can be extended easily to the multi-robot case, considering a decentralized approach. In this case, each robot will run locally Algorithm 1, updating its full-trajectory state with the information obtained from its sensors. When a robot i is within the communication range of other robot j, they can share their beliefs,

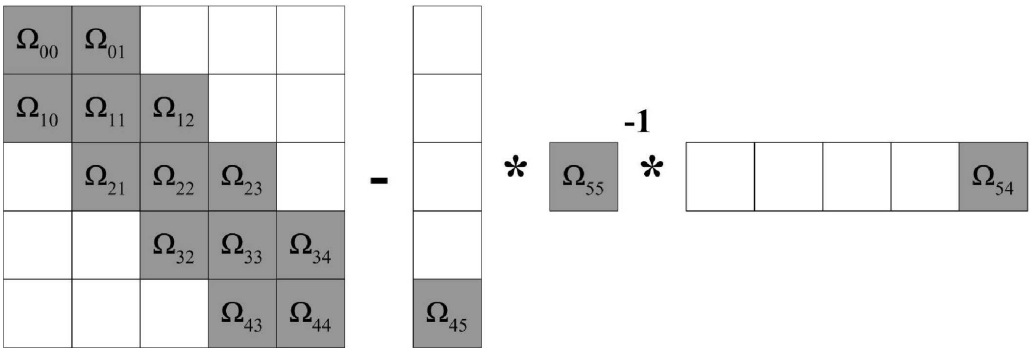


Figure 2.4: Marginalization of a removed point of the trajectory. Due to the structure of the information matrix, the marginalization only involves local block operations.

represented by their information vectors $\xi^i_{0:t}$ and $\xi^j_{0:t}$, and matrices $\Omega^i_{0:t}$ and $\Omega^j_{0:t}$. For Gaussian distributions, Equation (2.8) leads to a quite simple fusion rule:

$$\Omega^i_{0:t} \leftarrow \Omega^i_{0:t} + \Omega^j_{0:t} - \Omega^{ij}_{0:t} \tag{2.17}$$

$$\xi^i_{0:t} \leftarrow \xi^i_{0:t} + \xi^j_{0:t} - \xi^{ij}_{0:t} \tag{2.18}$$

which only requires using a separate EIF to maintain $\Omega^{ij}_{0:t}$ and $\xi^{ij}_{0:t}$ (which represent the common information exchanged between i and j in the past). It is important to remark that, if the IF constraints are fulfilled, using this fusion Equation and considering delayed states, the local estimator can obtain an estimation that is equal to that obtained by a centralized system as long as enough information is exchanged. Nevertheless, note that in the case of considering an EIF, local and centralized estimations are no longer exactly the same. This is because the Jacobians (linearization) calculated for each one could be evaluated for different points (μ_t) at certain time steps.

The common information can be locally estimated assuming a tree-shaped network topology (with no cycles or duplicated paths of information). However, this fixed network topology is a constraint too strong on the potential communication links among the (mobile) robots. If there are no assumptions about the network topology, before combining the beliefs, unknown common information should be removed. If

not, non-consistent estimations could be obtained due to the fact of adding several times the same information. Another option is to employ a conservative fusion rule, which ensures that the system does not become overconfident even in presence of duplicated information. As mentioned previously, for the case of the IF, there is an analytic solution for this, given by the Covariance Intersection algorithm (Julier and Uhlmann, 1997). Therefore, the conservative rule to combine the local belief of a robot i with that received from another robot j is given by:

$$\Omega^i_{0:t} \leftarrow \omega\Omega^i_{0:t} + (1-\omega)\Omega^j_{0:t} \tag{2.19}$$

$$\xi^i_{0:t} \leftarrow \omega\xi^i_{0:t} + (1-\omega)\xi^j_{0:t} \tag{2.20}$$

for $\omega \in [0\,1]$. It can be proved that the estimation is consistent (in the sense that no overconfident estimations are done) for any ω. The value of ω can be selected following some criteria, such as maximizing the obtained determinant of $\Omega^i_{0:t}$ (minimizing the entropy of the final distribution).

Although employing the CI formula avoids the need to maintain an estimation of the common information transmitted to the neighboring systems, as these fusion rules are conservative, some information is lost with respect to the purely centralized case.

Finally, Figure 2.5 depicts the scheme that follows the approach proposed here. The local DDSIF for a robot with its corresponding functional blocks is shown.

Synchronization of the trajectories

Special care has to be taken about synchronization issues when combining different trajectories. The trajectories are represented at discrete time intervals. The combination formula for two trajectories will work provided that the differences in the time steps of their intervals are bounded. Therefore, trajectories should be adjusted so that the state space is the same in both cases. Figure 2.6 depicts an example of the method proposed.

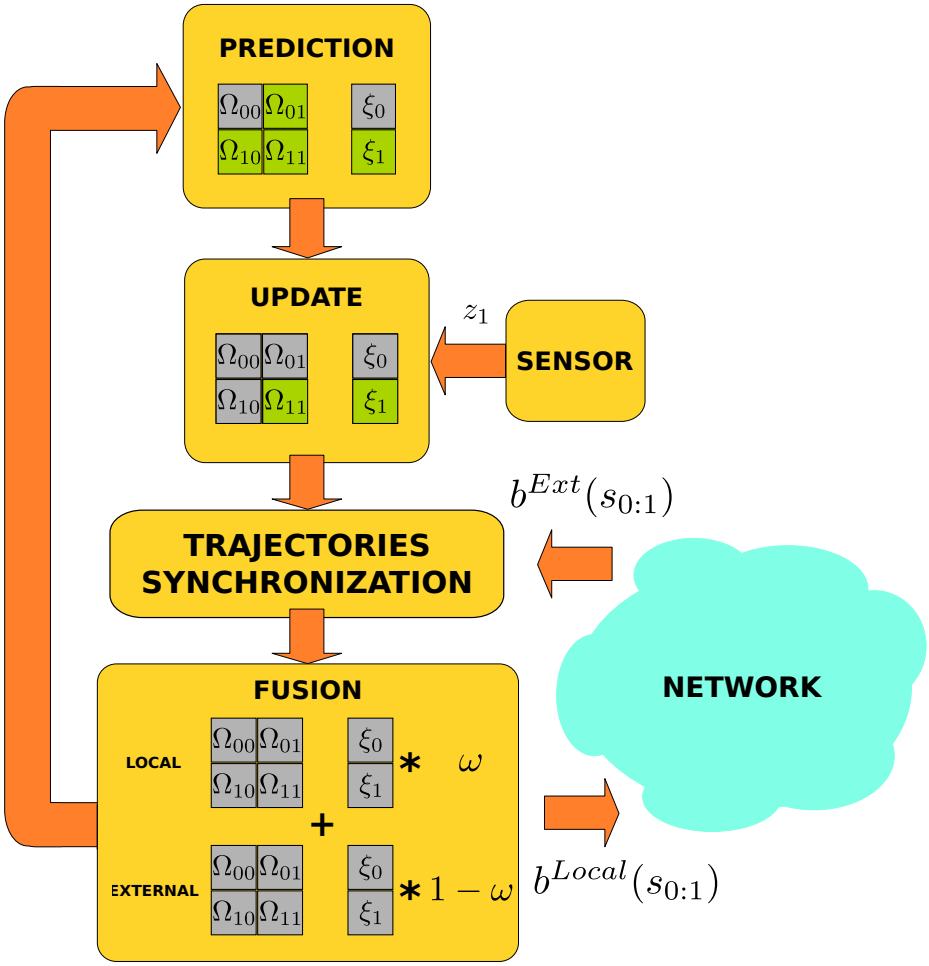


Figure 2.5: Flow chart of the local DDSIF proposed for each robot. For simplicity, only an example with trajectories of two time instants is shown. In the prediction and update stages, the blocks of the information matrix and vector that are modified are colored green.

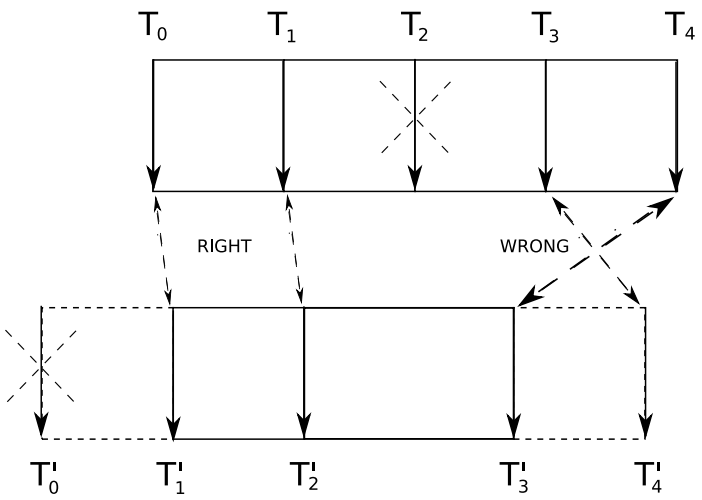


Figure 2.6: Example of the method to synchronize two trajectories.

In this method, first, the newest time steps are predicted by using Equations (2.12) and (2.13), and the oldest ones are marginalized out (2.16) until trajectories are adjusted. Thus, in the example, T_0' must be removed and T_4' predicted. Then, each time step is matched with the closest one of the other trajectory. Furthermore, matchings are just allowed if the time difference is lower than a certain threshold. Time steps without matching must also be marginalized out before fusing (T_2 in the example).

Finally, notice that the algorithm cannot allow crossed matchings such as the one labelled as *WRONG* in Figure 2.6 This kind of wrong matchings could result in serious errors in the estimations.

2.6 Simulated example

In this Section, some simulations in Matlab are shown. These simple examples were simulated in order to show the concept of missing information when full trajectories are not considered in the estimation of the state. Similar examples are shown by Nettleton (2003) for the same purposes.

The simulations consist of two robots with sensors tracking a moving vehicle, which is able to move along the x axis (see Figure 2.7). The state to estimate is

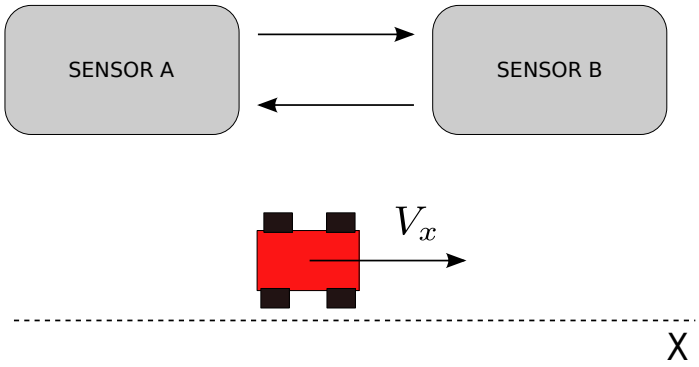


Figure 2.7: Example simulated in Matlab.

composed by the position x in meters, and the velocity v_x in meters per second of the vehicle at every time step:

$$s_t = \begin{pmatrix} x & v_x \end{pmatrix}^T \tag{2.21}$$

Each robot has a noisy sensor that can measure the vehicle's position directly, so the update model used is linear:

$$z_t = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot s_t + \varepsilon_t \tag{2.22}$$

with the noise variance $S_t = 1m^2$. The simulated vehicle always starts at $x = 5m$ and moves with a constant velocity $(8m/s)$. As a generic motion model of the target, a discrete version of a continuous white noise acceleration model or second-order kinematic model is used (Bar-Shalom et al., 2001; Stone et al., 1999). In this model, the velocity is assumed to be affected by an acceleration modeled as a white noise of zero mean and with power spectral density q. The discrete version of this linear motion model is characterized by:

$$A_t = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \tag{2.23}$$

and

Robot 1		**Robot 2**	
Measurement	Time step	Measurement	Time step
46.18	5	165.91	20

Table 2.1: Simulated measurements for the experiment 1

Robot 1		**Robot 2**	
Measurement	Time step	Measurement	Time step
46.18	5	84.41	10
205.63	25	125.12	15
		165.91	20

Table 2.2: Simulated measurements for the experiment 2

$$R_t = \begin{pmatrix} \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & \Delta t \end{pmatrix} q \tag{2.24}$$

where $\Delta t = 1s$ and $q = 0.05m^2/s^3$.

Two similar experiments with the same two robots tracking the vehicle were performed. The measurements gathered locally by each robot's sensors are summarized in Tables 2.1 and 2.2. In these examples, the measurements are positions in meters and each time step corresponds to 1 second.

For every experiment three different approaches were run: (i) a centralized IF, (ii) a decentralized IF without considering delayed states (that is, the proposed algorithm but considering only the last state), and (iii) the proposed delayed-state IF. Moreover, since there were just two pursuers, a Channel Filter was able to compute exactly the common information between the two perception entities. Thus, no CI rules were needed during the performance.

The filter initialization was the same for all the simulations and both robots:

$$\begin{aligned} \mu_0 &= \begin{pmatrix} 5 \\ 10 \end{pmatrix} \\ \Sigma_0 &= \begin{pmatrix} 2.5 & 0 \\ 0 & 3 \end{pmatrix} \end{aligned} \tag{2.25}$$

The values shown in the Tables 2.1 and 2.2 were obtained by simulating the previous models of the vehicle and the sensors for 25 seconds. In both experiments, at the time step 25 the robot 1 fused information received from the other one. The measurements of the sensors were taken at previous time steps in order to show what happens when the fusion is not done every time a local measurement is updated. Thus, between the initialization (which is the same for both robots) and the fusion step, there are several local predictions and updates that are not transmitted until the end of the experiment, what leads to some differences between the tested approaches.

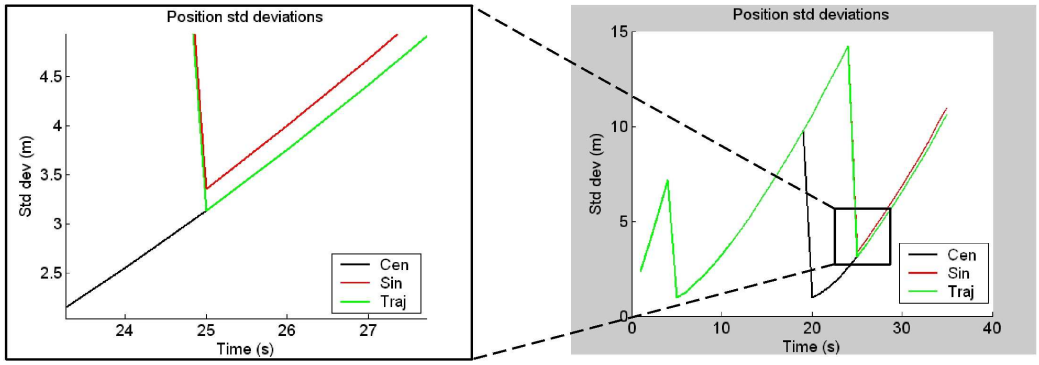
Figure 2.8 depicts the results of the simulations. Regarding the vehicle's position, standard deviations of the three approaches are compared for the experiment 1 and 2. For simplicity, only the results for the robot 1 are shown. In both experiments, a zoom has been made at the time step 25 in order to highlight the differences after the fusion.

The specific values of experiment 1 try to show how, even in such a simple example, the centralized solution is not recovered with the decentralized filter which does not consider the full-state trajectories. However, the delayed-state filter is able to recover this centralized estimation. Besides, experiment 2 is included to show another remarkable result: the estimation without delayed states can become overconfident (its variance is smaller than the centralized one after the fusion). This fact indicates that, without considering the trajectories, the estimation could become inconsistent in some cases.

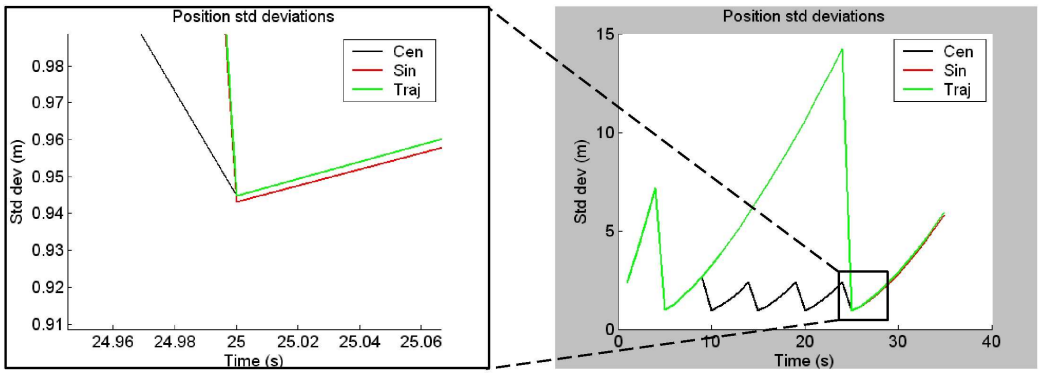
These simulations intend to be a proof of concept for the delayed-state filter proposed. In the next Chapter, extensive results will be shown in order to prove the performance of the approach with real teams of robots and sensors.

2.7 Conclusions

The Chapter presented a decentralized data fusion scheme valid to perform cooperative active perception tasks using a set of heterogeneous sensors. An extension of the usual EIF considering delayed states was proposed, which obtains locally the same



(a)



(b)

Figure 2.8: Standard deviations for the experiment 1 (a) and the experiment 2 (b). Black solid line is the centralized estimation, the red solid line is the decentralized estimation without considering delayed states and the green solid line refers to the proposed delayed-state IF. On the left hand side, both experiments are zoomed in at the fusion instant to show that the delayed-state IF is superimposed on the centralized solution.

estimation as a centralized filter, and makes easier to overcome the usual delays and breakdowns in inter-process communication.

In addition, methods to match trajectories from different filters and to fuse the information in a conservative way were explained. This is particularly important in decentralized architectures in order to face the problem of double-counting information.

Simple simulations were proposed in order to illustrate how some information may be missed with respect to the centralized case when a standard IF is used in a decentralized manner. The presented approach was proved to be valid to overcome this problem.

The proposed method is attempted to be a generic approach. Hence, some approximations were described so that non-linear models could be considered and no constraints about the network topology had to be made. It should be noticed that when the information is fused in a conservative manner or linearized models are used, the centralized solution is no longer reached.

Nevertheless, considering state trajectories can still be a powerful tool for other purposes and it improves the robustness of the system. For instance, maintaining delayed states, wrong data associations made in the past could be detected later and fixed by recomputing the trajectory up to the current time step. Besides, in certain applications, such as tracking, recording the trajectory of the target would provide more useful information when a prediction of its movement has to be made before planning actions.

In addition, another advantage of the Delayed-State Information Filter is that the communication bandwidth is increased at an affordable rate and the trajectories can be bounded in time by using the presented algorithms. The length of these trajectories must be chosen to reach a compromise between missing information and required bandwidth.

Chapter 3

Experimental results in cooperative tracking applications

In order to test the decentralized perception scheme presented in Chapter 2, a tracking application is considered in this Chapter. The method will be applied to cooperative tracking in two different environments: a disaster management scenario and an urban scenario. In both of them, a potential target will move along outdoor and indoor areas, so the fusion of data provided by heterogeneous sensors is beneficial.

All of the results presented in this Chapter were obtained during the experimental sessions of the European projects AWARE and URUS. The experimental setups of both projects and their results in cooperative tracking are also shown.

3.1 Experimental setup of the AWARE project

The AWARE project and its objectives were introduced in Chapter 1. The project included the development of an architecture for the cooperation of heterogeneous sensors and robots, both aerial and ground. To verify the success in reaching the objectives, the AWARE project considered the validation in Disaster Management/Civil Security applications. In particular, some of its functionalities were cooperative tracking of fire-fighters and cooperative fire monitoring.

An experimental scenario was installed in the facilities of the Protec-Fire company (Iturri group) located in Utrera (Spain). A structure was used to emulate a building where an emergency had occurred. During the experiments, fire and smoke machines were used to produce controlled fires in the building. In the surroundings of the building, fire-fighters were allowed to operate.

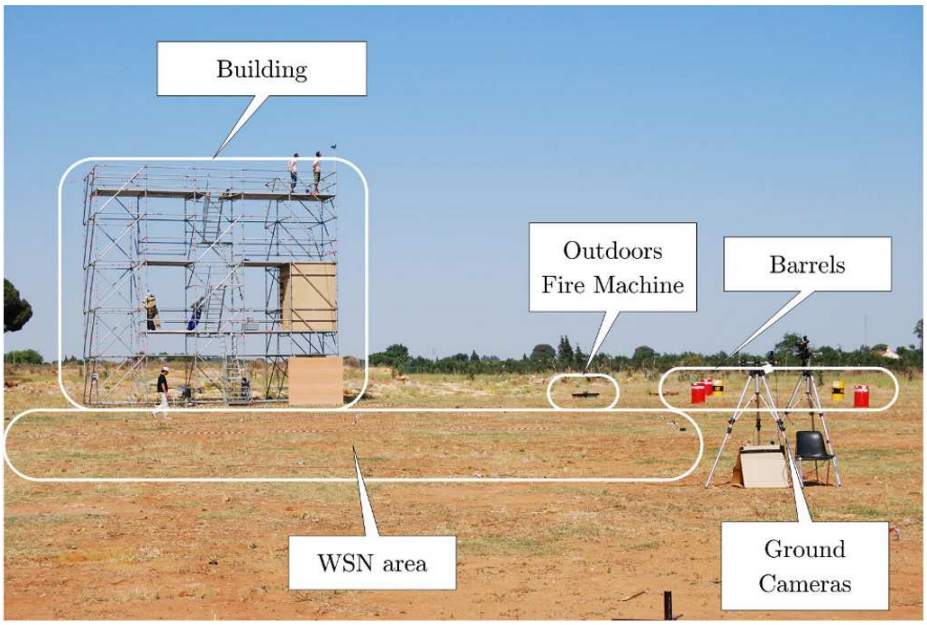
Three general experiments, one per year of the project, were conducted in a common scenario (see Figure 3.1). The particular components integrated in the AWARE platform during the experiments were the following:

- three Unmanned Aerial Vehicles (UAVs) equipped with visual/infra-red cameras.
- a network of ground cameras mounted on tripods at known positions. In particular, two visual cameras to monitor the area of interest and an infra-red camera to monitor possible fires in the building.
- a Wireless Sensor Network (WSN) equipped with different types of sensors (temperature, humidity, CO, smoke, etc.) and with mobile nodes. This network made possible the detection of potential alarms, such as a fire in the area. If a fire-fighter carried one of the mobile nodes, information about its location could also be obtained.

3.2 Real experiments in the AWARE project

Cooperative tracking is one of the functionalities of the AWARE project, and the algorithms presented in Chapter 2 were applied within the framework of this project in order to track real fire-fighters in an outdoor disaster management scenario.

The AWARE platform is composed of different Perception Sub-Systems (PSS) cooperating among them in order to achieve common objectives (see Figure 3.2). In Chapter 2, each robot was modeled as a perception node (with its own sensors) integrated into a network that estimated the state of the environment in a decentralized fashion. Here, each PSS represents one of these perception nodes and runs a perception process (an instance of the decentralized filter in Chapter 2) which is in charge of



(a)

(b)

(c)

Figure 3.1: (a) Situation of the different components of the experimental setup during the AWARE experiments. (b) UAVs flying over the experimental area. (c) Ground cameras used to gather visual information during those experiments.

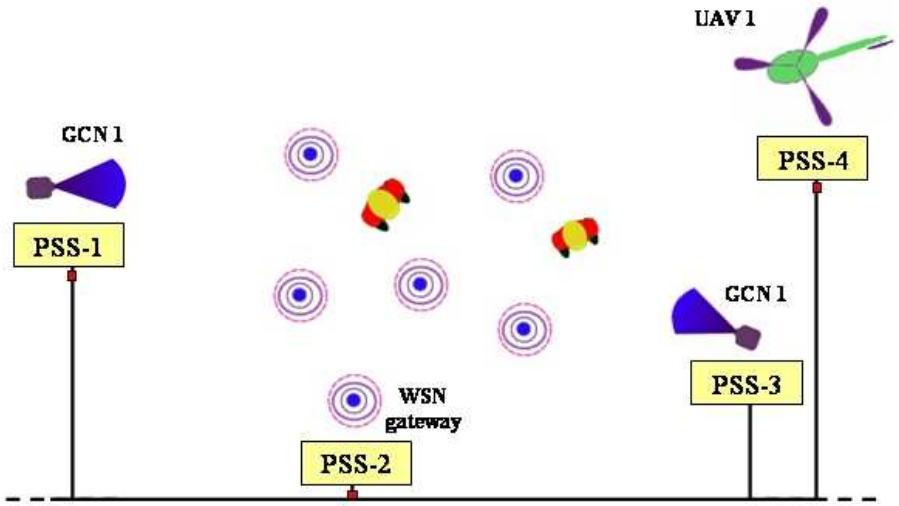


Figure 3.2: AWARE sub-systems. The UAVs carry the aerial cameras, whereas GCN stands for Ground Camera Network and WSN for Wireless Sensor Network.

updating the sub-system status. Thus, this process incorporates the local information obtained by its sensors as well as the information provided by other PSSs (neighbors' beliefs).

Experimental results obtained during real field experiments integrating three sources of information (ground cameras, aerial cameras and a wireless sensor network) are presented. The information provided by these sensors was used to track the position of a fire-fighter moving around the experimental area by means of the decentralized data fusion approach.

A C++ implementation of the decentralized data fusion scheme called Perception Sub-System (PSS) was used to process locally the data gathered by each sensor. These processes performed a prediction/update step every 0.2 seconds and included measurements from local sensors as well as the information provided by other PSSs (neighbors' beliefs). In these experiments, there were only two different kinds of PSSs, since two types of sensors were used: visual cameras and the WSN. Moreover, cameras provided local measurements at 1 Hz to 3 Hz approximately, while the WSN updated local measurements at 10 Hz. By default, beliefs were communicated to other neighbors only when new local measurements were incorporated. A PSS for

each perception entity (either camera or WSN) was launched during the experiments. The models for the different sensors and for the prediction of the state are detailed in the next Sections.

3.2.1 State prediction

The state estimated and shared between PSSs consists of the 3D position and velocity of the person to track, both expressed in a global coordinate system:

$$s = \begin{pmatrix} x & y & z & v_x & v_y & v_z \end{pmatrix}^T \tag{3.1}$$

The prediction model is the same second-order kinematic model (Bar-Shalom et al., 2001; Stone et al., 1999) that was used in the examples of Chapter 2, where the acceleration was modeled by a white noise. In this case, $\Delta t = 0.2s$ and $q = 0.05m^2/s^3$:

$$A_t = \begin{pmatrix} I & \Delta t I \\ 0 & I \end{pmatrix} \tag{3.2}$$

and

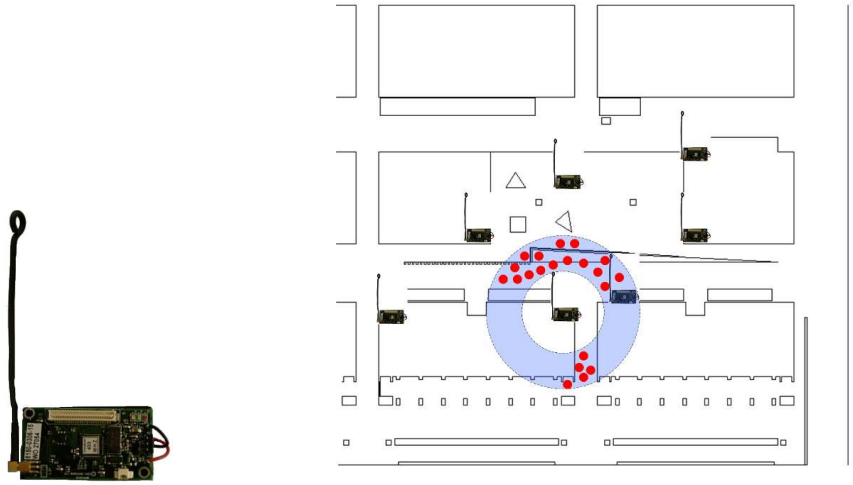
$$R_t = \begin{pmatrix} \frac{1}{3}\Delta t^3 I & \frac{1}{2}\Delta t^2 I \\ \frac{1}{2}\Delta t^2 I & \Delta t I \end{pmatrix} q \tag{3.3}$$

3.2.2 Measurements from the WSN

When dealing with heterogeneous PSSs, different measurement models have to be used for each type of sensor. In this case, the whole WSN acts as a single sensor with a PSS associated. In particular, the network is composed of Mica2 nodes[1] (see Figure 3.3(a)) that are able to sense different quantities, such as pressure, temperature, humidity, etc. Moreover, they have wireless communication devices, and they are able to form networks and relay the information that they gather to a gateway.

In addition, the signal strength (Received Signal Strength Indicator, RSSI) received by a set of static nodes whose positions are known can be used to infer the

[1]These nodes are provided by the manufacturer Crossbow (http://www.xbow.com).



(a) (b)

Figure 3.3: (a) Mica2 node. (b) Particles (red) are used to represent person hypotheses. The signal received by a set of static nodes can be used to infer the position of the node. The filter is initiated when the first message is received by sampling uniformly from a spherical annulus around the receiver. Map information is also taken into account (only free spaces within the annulus are considered).

position of a mobile object or person who is carrying one of the nodes (the emitter). The algorithm to estimate and track the node position is based on particle filtering. In the particle filter, the current belief about the position of the mobile node is represented by a set of particles $\{p_t^{(i)}\}$ which represent hypotheses about the position of the person that carries the node at time t (see Figure 3.3(b)).

In each iteration of the filter, kinematic models of the person motion and map information are used to predict the future positions of the particles. The likelihood of these particles is updated any time new messages are received from the static network. The technique is summarized in Algorithm 2, where o_t^j is the measurement provided at time t by a static node j, consisting of its position p^j and the strength of the received signal from the mobile node, $RSSI_t^j$. Next subsections describe further the main steps in this Algorithm.

Algorithm 2 $\{p_t^{(i)}, \omega_t^{(i)}; i = 1, \ldots, L\} \leftarrow$ Particle_filter($\{p_{t-1}^{(i)}\omega_{t-1}^{(i)}; i = 1, \ldots, L\}, o_t^j = \{p^j, RSSI_t^j\}$)

1: **for** $i = 1$ to L **do**
2: $p_t^{(i)} \leftarrow$ sample_kinematic_model $(p_{t-1}^{(i)})$
3: **end for**
4: **if** Message from network o_t^j **then**
5: **for** $i = 1$ to L **do**
6: Compute $d_t^{(i)} = \|p_t^{(i)} - p^j\|$
7: Determine $\mu(d_t^{(i)})$ and $\sigma(d_t^{(i)})$
8: Update weight $\omega_t^{(i)} = p(RSSI_t^j|p_t^{(i)})\omega_{t-1}^{(i)}$ with $p(RSSI|p_t^{(i)}) = \mathcal{N}(\mu(d_t^{(i)}), \sigma(d_t^{(i)}))$
9: **end for**
10: **end if**
11: Normalize weights $\{\omega_t^{(i)}\}, i = 1, \ldots, L$
12: Compute $N_{eff} = \dfrac{1}{\sum_{i=1}^{L}(\omega_t^{(i)})^2}$
13: **if** $N_{eff} < N_{th}$ **then**
14: Resample with replacement L particles from $\{p_t^{(i)}, \omega_t^{(i)}; i = 1, \ldots, L\}$, according to the weights $\omega_t^{(i)}$
15: **end if**

Prior prediction and importance functions

Regarding the prior function, the option used is to initialize the filter with the first message received from the mobile node, by considering a uniform distribution on a spherical annulus around the receiver. The map of the scenario, if available, is taken into account when sampling from this prior (see Figure 3.3(b)), by assuming that the person cannot be inside any building or forbidden place.

Each time step, the positions of the particles are predicted from their previous positions (line 2 of Algorithm 2). The prediction function uses the same Brownian motion model as in the previous Information Filters (Stone et al., 1999). This model is combined with map information to discard unfeasible motions (like going through walls); particles arriving at occupied places are rejected and substituted by new sampled particles.

The likelihood function

The likelihood function $p(RSSI_t|p_t)$ plays a very important role in the estimation process, since each time a message is received this likelihood is used to update the particles weights (lines 5 to 9). This likelihood models the correlation that exists between the distance that separates two nodes and the $RSSI$ value. This correlation decreases with the distance between the two nodes, transmitter and receiver (Caballero et al., 2008). This is mainly caused by radio-frequency effects such as radio reflection, multi-path or antenna polarization.

The model used here considers that the conditional density $p(RSSI_t^j|p_t)$ can be approximated as a Gaussian distribution for a given distance $d_t^j = \|p_t - p^j\|$ between the mobile node and a static node j, as follows:

$$RSSI_t^j = \mu(d_t^j) + \mathcal{N}(0, \sigma(d_t^j)) \tag{3.4}$$

where the mean $\mu(d_t^j)$ and the standard deviation $\sigma(d_t^j)$ are non-linear functions of the distance (which itself is a non-linear function of the state). These functions are estimated during a calibration procedure. Figure 3.4 shows the experimental data used in a calibration, and the estimated relations.

Measurement model

The person being tracked carried a wireless sensor node that was used by the WSN to provide position information based on the previous Algorithm. The deviation of these measurements was provided by the localization Algorithm and depended on the density of the WSN deployed. In the case of the AWARE experiments, the deviation ranged from 3 to 10 meters for x and y axis. Since the output provided by Algorithm 2 consisted of poses referenced to a global frame, it was also used to initialize the track.

The wireless sensor network (see Algorithm 2) provides a probability distribution (particles) over the position of the person. Once this distribution has converged to a uni-modal distribution, it can be assumed to be a Gaussian with mean $\mu(p_t)$ and covariance matrix $\Sigma(p_t)$. Since this information is referenced to the global coordinate

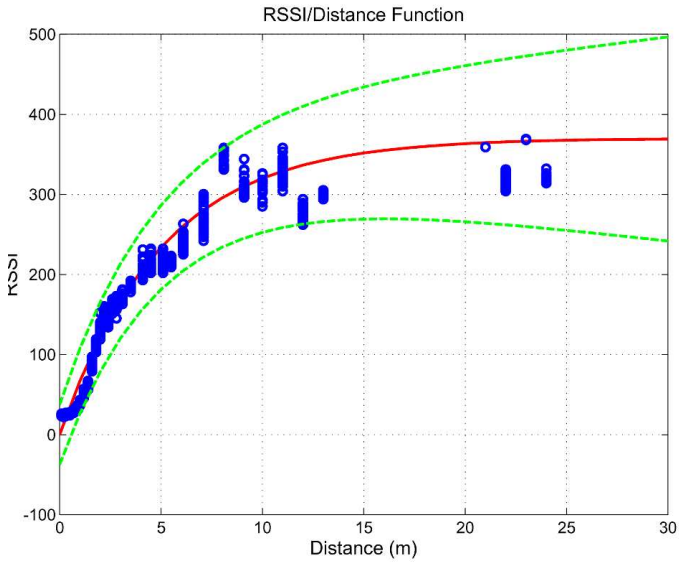


Figure 3.4: RSSI-distance functions, $\mu(d)$ and $\sigma(d)$. These functions relate the distance between two nodes and the RSSI received in mean and standard deviation. They have been computed experimentally by using a large set of RSSI/distance pairs. The RSSI representation is the one used in the Mica2 nodes, 0 is the maximum signal strength and 375 the minimum. Dots: A sub-set of the experimental set of data. Solid line: Estimated mean $\mu(d)$. Dashed lines: standard deviation confidence interval based on $\sigma(d)$.

system, its relation with the state in Section 3.2.1 is straightforward, and it can be used as a measurement to estimate that state:

$$z_t^{wsn} = \begin{pmatrix} x_t^{wsn} & y_t^{wsn} & z_t^{wsn} \end{pmatrix}^T = \mu(p_t) \tag{3.5}$$

3.2.3 Measurements from the cameras

In addition, the cameras were used to detect fire-fighters into their field of view, providing bearing-only information about the position. These measurements can be modeled by a non-linear pin-hole model. For all the cameras, the intrinsic and extrinsic calibration parameters were assumed to be known.

(a) (b) (c)

Figure 3.5: Samples of the output of the visual detector. The images are taken from two different fixed cameras (right, left) and a camera mounted on a UAV (middle). The centers of the red ellipses indicate the measurements of the detector.

Image processing was applied to detect new candidates in the image by using background learning techniques, and this information was matched with a color segmentation output to build blobs of homogeneous color around each cluster (Forssén, 2004). Finally, a classifier was used to differentiate between fire-fighters and other objects by using information and scores obtained from blobs and color clusters (Capitan et al., 2007). Some examples of the detector output are shown in Figure 3.5. The uncertainty associated with the person measurements provided by this approach (they are expressed in pixel coordinates u and v on the image plane) are approximately $\sigma_u = 15$ and $\sigma_v = 12$ when images with a resolution of 640×480 pixels are considered. To avoid false positive detections, the image detector makes use of the RSSI estimation (which is very noisy but with a very low rate of false positives) to discover and remove outliers. This is only done in the initialization of the image processing algorithm.

Therefore, each camera provides a measurement composed of the position (u, v) and velocity $(\dot{u}, \dot{v})$ of the target referenced to its image plane, expressed in pixel and pixel/s, respectively.

$$z_t^{cam} = \begin{pmatrix} u_t^{cam} & v_t^{cam} & \dot{u}_t^{cam} & \dot{v}_t^{cam} \end{pmatrix}^T \tag{3.6}$$

In order to obtain the measurement model, it is needed to relate objects on the image plane with their positions in the 3D world. Cameras project points in the

space into points on the image plane and are usually modeled by using the tools of projective geometry (Hartley and Zisserman, 2004; Faugeras and Luong, 2001). This projection can be modeled as a *pin-hole* projection. Following this model, each point in the space, p and its corresponding image pixel m on the image plane of the camera are related by Equation (3.7), where p and m are given in homogeneous coordinates:

$$\eta \cdot m = K \begin{pmatrix} Rot & -Rot \cdot p_{cam} \end{pmatrix} p \tag{3.7}$$

Here, K is the upper triangular intrinsic calibration matrix of the camera; Rot is the rotation matrix from the global reference system to the camera system; p_{cam} is the translation camera vector in the global system; and η is a scale constant.

Previous Equations imply a non-linear relation between the state and the measurements (due to the homogeneous coordinates). Actually, if the camera pose is uncertain, it has to be considered when obtaining the corresponding likelihood (and, also in this case, the relation among variables are non-linear). Therefore, if an IF is to be used, a previous linearization is required. In this case, a first order Taylor expansion was used in order to derive an EIF. Although camera poses at each step were assumed to be known, their uncertainties were also considered and propagated through the Jacobians of the models. In such a way, the noise vector was composed by the additive noises from the measurement itself (they depended on the segmentation algorithm accuracy) and the camera pose uncertainties.

It can be seen that the measurements obtained by the cameras and the wireless sensor networks depend only on the position and velocity of the target (the state), and thus the conditional independence assumption among measurements of Chapter 2 is applicable here.

3.2.4 Data association

Regarding the data association problem, some considerations have to be done. For the measurements from the WSN, since each moving node reports its identifier, the data association is straightforward. In the case of the cameras, local data association is solved by projecting the state estimation on the image plane and computing the

Mahalanobis distance to the measurements. Finally, note that the WSN was used to initialize the estimations of the objects of interest and assign univocal identifiers to them (wireless nodes identifiers). Thus, all the objects had the same identifier at each sub-system, removing the need to deal with data association among them.

3.2.5 Experiment 1

Three different experiments are presented with the AWARE platform. In all of them, the results of the proposed decentralized algorithm are compared with the results obtained by a centralized implementation. In that version, all the measurements were processed offline by a centralized EIF without considering communication issues or delays. The centralized filter has access to all the information provided by all the sensors instantly, and it is not affected by double counting of information, asynchronous data or partial information. A comparison of the estimations of the decentralized perception system with a centralized filter provides a good measure of the efficiency of the proposed approach. In addition, it indicates how far the decentralized approach is from the optimal centralized filter, which is proposed as ground-truth.

Particularly, this Section presents an experiment integrating two ground cameras and a WSN. The estimated x, y and z positions of the target provided by the filter attached to camera 1 are shown in Figure 3.6. It can be seen how the error with respect to the centralized estimation is, in mean, about one meter. Besides, the estimated standard deviation of the filter is coherent with the errors with respect to the centralized estimation (ground-truth), which is always inside the 3σ confident interval.

Note that in this case, thanks to the fusion between different sources, the bearing-only information provided by the cameras can be used to estimate the full target position. This issue would have been hardly addressed with a single camera.

Another important aspect in decentralized approaches is to verify that the estimations obtained by the different filter instances converge to a single solution. This is shown in Figure 3.7, where the estimated xy trajectory provided by camera 1, camera 2 and the wireless sensor network are plotted together with the centralized

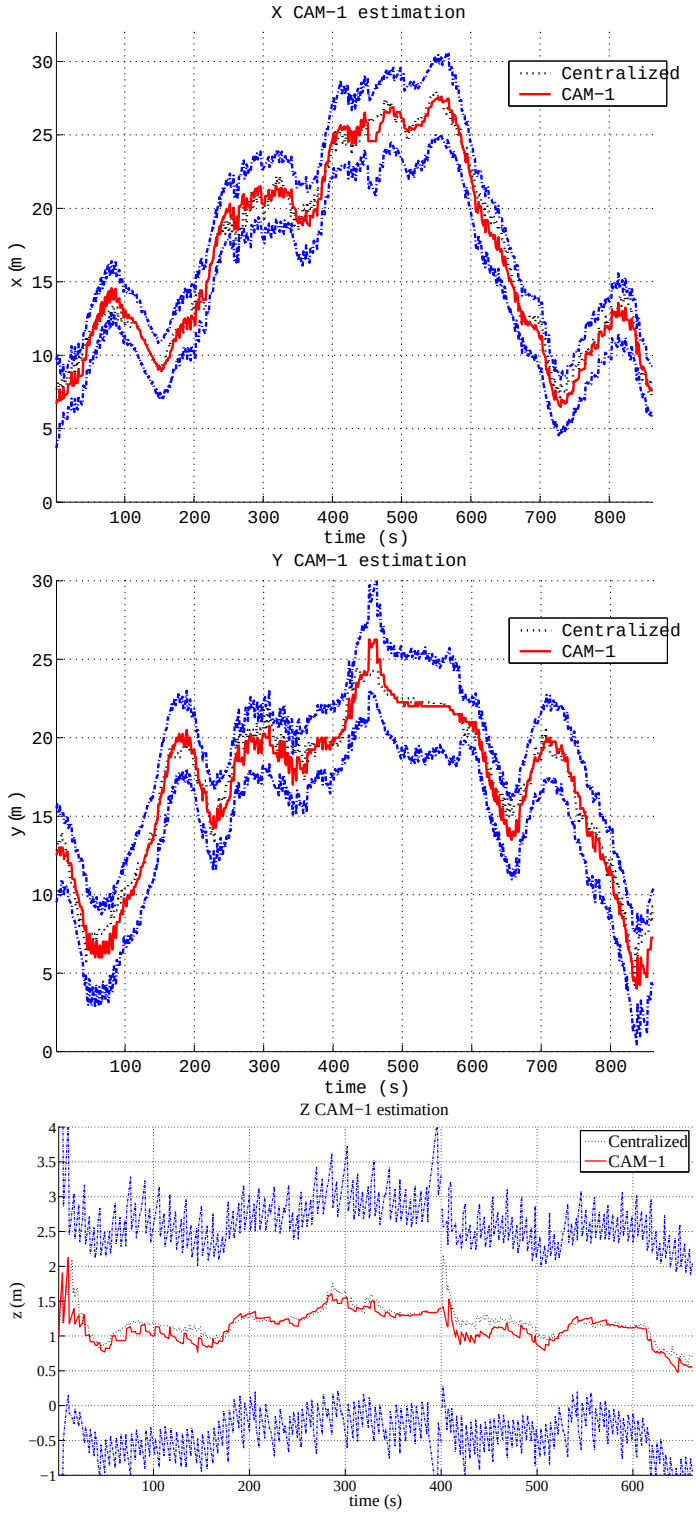


Figure 3.6: Position estimation of the person using the decentralized approach presented in Chapter 2 (red solid line) in camera 1. The estimation provided by a centralized filter is also presented (black dotted line). It can be seen how the centralized estimation is always inside the 3σ confident interval (blue dashed line).

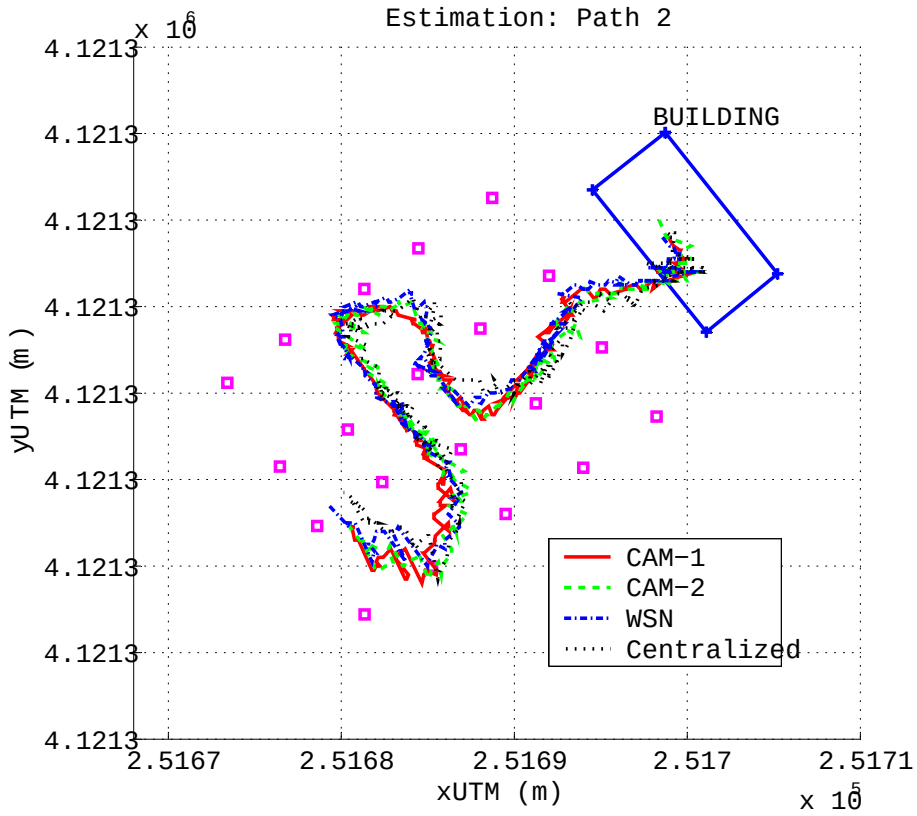


Figure 3.7: *xy* estimation provided by the two cameras and the wireless sensor network, and centralized estimation. A sensor network was deployed into the experimental area, pink squares denote the position of each sensor node.

estimation. It can be seen how all of the estimations converge to the same solution with errors in the order of one meter.

The estimated standard deviation computed by the decentralized approach and the estimated by the centralized filter are presented in Figure 3.8. The decentralized approach presents more conservative estimations than the centralized filter. The difference between the solutions in this case is explained by different linearization points for the Jacobians in the centralized and decentralized filters; and, mainly, by the use of the *covariance intersection* algorithm. During all the experiments in AWARE, this conservative fusion rule was employed, since the network topology had to be flexible as a project requirement (no tree-like topologies were possible). The

criterion to select the value of ω parameter was to use a fixed weight that setups the system confidence in its own estimation and the neighbor's ones.

However, it is worth to mention the closeness of the decentralized and centralized estimations, which differ in no more than half a meter. This fact remarks the consistency and benefits of the proposed approach. In general, the standard deviation grows during the prediction steps and decreases after updating with local or external information, showing a high-frequency component. Since the centralized filter is accessing all of the information at every time step, it is updated more often, presenting a lower variation over the mean standard deviation.

In general, the experiments showed that the proposed decentralized approach is able to provide estimations with small errors (one meter) with respect to centralized filters and very similar standard deviation estimations (less than half a meter difference), but with the advantage of processing the information in a fully decentralized manner, which basically improves the fault tolerance and scalability of the system.

3.2.6 Experiment 2

This Section presents an experiment to show how the algorithm works when the transmission frequency between the fusion nodes is increased. Results can be seen in Figure 3.9. In these experiments, with the same three sources of information that were used above, the transmission frequency was varied from 1 second to 5 seconds. It can be seen how, after the fusion steps, the estimation is very similar in all cases due to the use of delayed states. Clearly, between the fusion steps, the differences are higher when the frequency is lower (however, after the fusion steps, the past states are also recovered, and hence the same solution as in the central filter is achieved, although with latency). This is an expected result and it indicates that the transmission frequency is a parameter that should be chosen as a compromise between performance and required bandwidth. Considering the network facilities and the maximum communication delays for the AWARE/URUS setups, in all the experiments of this Chapter, 5-second trajectories were used so that no information

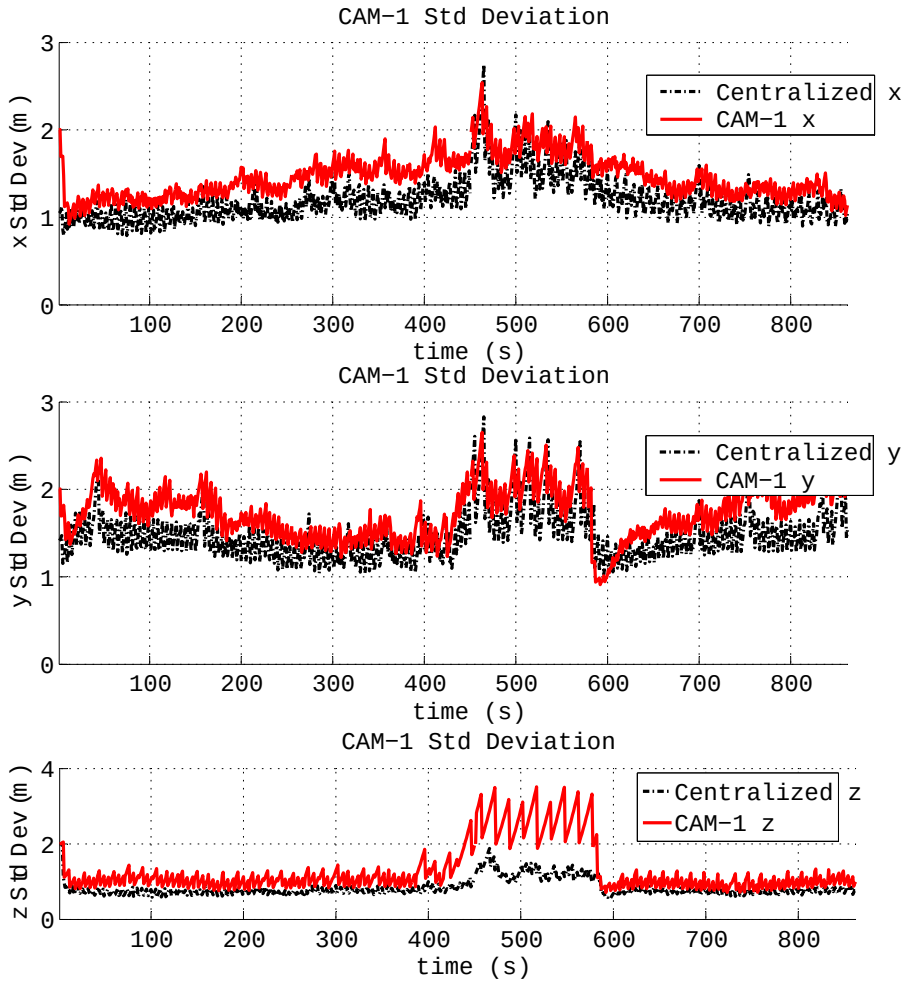


Figure 3.8: Estimated standard deviation using the decentralized approach (red solid line) versus standard deviation computed by the centralized filter (black dashed line).

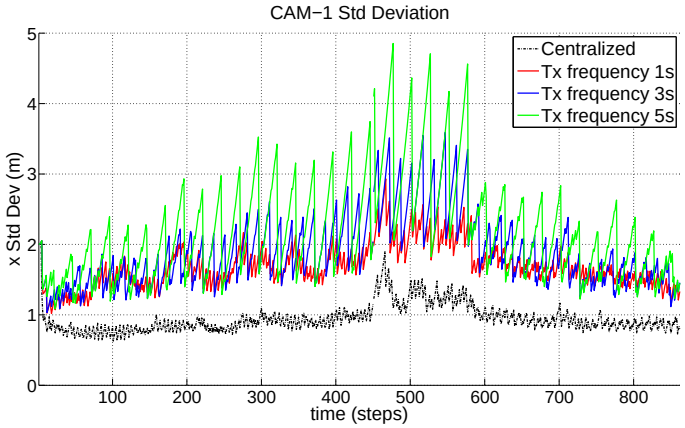


Figure 3.9: Estimated standard deviation using the decentralized approach varying the transmission latency versus standard deviation computed by the centralized filter.

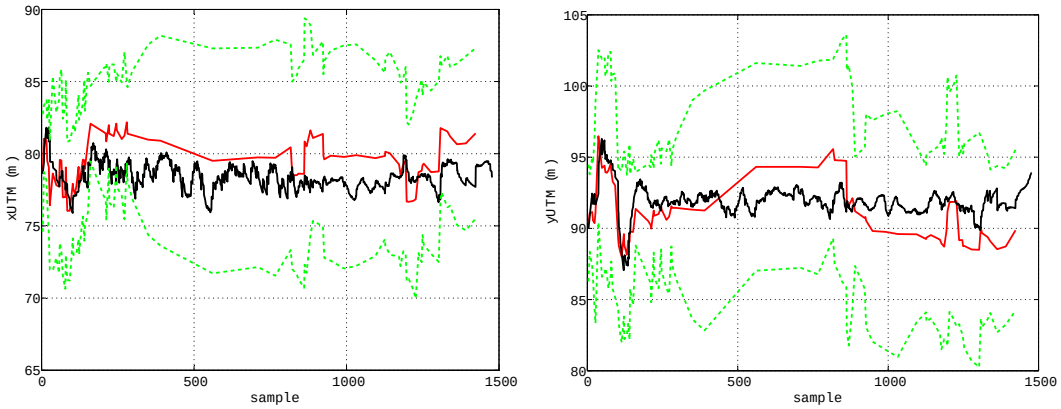
was missed during the performance. Again, differences with respect to the centralized estimation are mainly due to the CI algorithm.

3.2.7 Experiment 3

Finally, this experiment contains some figures in order to provide a clear picture of the sensor capabilities used in this platform and how the proposed decentralized data fusion approach benefits the global estimation. In particular, an experiment integrating a WSN and two aerial cameras is presented.

The estimated position of the fire-fighter based only on the WSN is shown in Figure 3.10. This estimation comes from the received signal strength (RSSI) provided by the sensor node that the fire-fighter carries. It can be observed that the sensor is very noisy, with an average error between 2 and 3 meters. It is important to mention that until sample 100 the centralized filter (used as ground-truth) only integrated information from the received signal strength filter, and for this reason the centralized filter matches very well with the RSSI sensor in the first samples.

Although noisy, the RSSI-based estimation provides valuable information to the system. It becomes critical when bearing-only estimations are considered, like the one from a single camera. The absence of scale factor could lead the filter to diverge if the

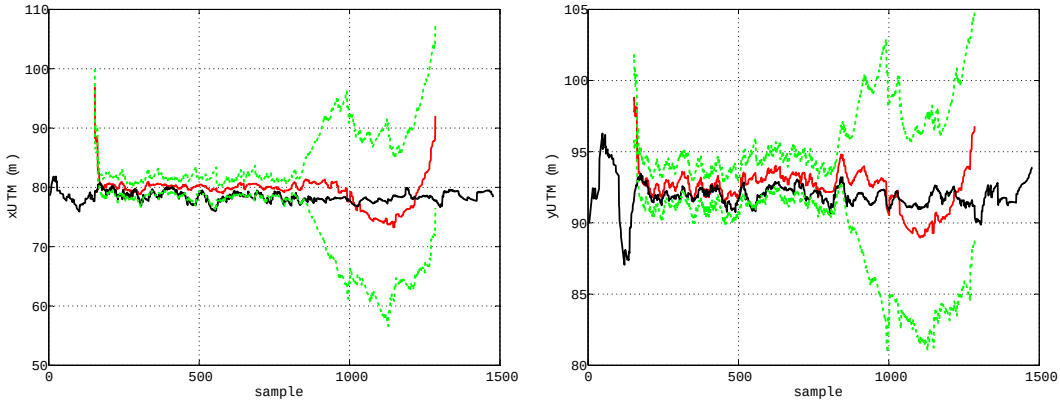


(a) (b)

Figure 3.10: Estimated x and y positions of the fire-fighter based on the received signal strength provided by the wireless sensor network. Red: Estimation provided by the received signal strength filter. Green: Estimated 3σ confidence interval. Black: Centralized estimation with all the sensors used as ground-truth.

camera does not triangulate enough with respect to the target position. The RSSI-based estimation intrinsically provides a bound on the scale of those cameras. This benefit will always be produced when absolute estimations (RSSI-based, beacons, GPS, etc.) are integrated into the global estimation.

The estimated position of the fire-fighter based only on the images from the cameras on board the helicopters is represented in Figure 3.11. The single estimation of each helicopter is not shown because they are bearing-only estimations, so triangulation from a different position is needed. The figure shows how the helicopters began the estimation at about sample 100 and how this estimation is good until sample 800 approximately, at which point one of the helicopters moved to a position where the fire-fighter was out of the field of view and the estimated position diverged due to the lack of range information with a single camera. It is easy to see that the estimation would be improved if another source of information, such as a ground camera or the received signal strength localization were integrated with the helicopter's information. In fact, the decentralized data fusion approach presented in this Thesis integrates all the data sources automatically, taking into account their associated uncertainties and achieving a consistent global estimation better than that provided by isolated sensors.



(a) (b)

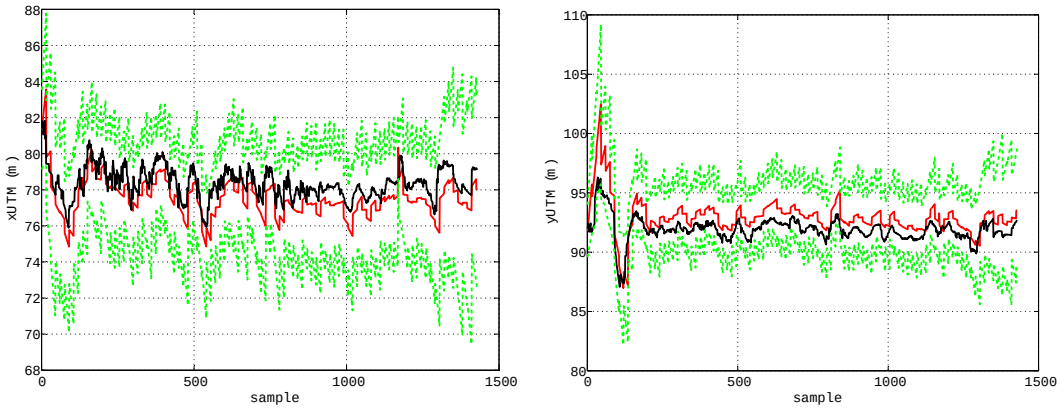
Figure 3.11: Estimated x and y positions of the fire-fighter based on the images gathered by the UAVs. Red: Estimation provided by a centralized filter integrating the images of both helicopters. Green: Estimated 3σ confidence interval. Black: Centralized estimation with all of the sensors used as ground-truth.

In Figure 3.11 is also shown how the estimation provided by the UAVs is accurate when compared to the centralized filter. The optimal positions at which the UAVs took the images of the fire-fighter are very relevant to explain these good results. These positions where computed online during the experiment to keep the fire-fighter within the field of view.

Finally, the estimation of the fire-fighter position by means of the decentralized approach fusing information from all the sources (two aerial cameras and the WSN) is shown in Figure 3.12 for comparison.

3.3 Experimental setup of the URUS project

In order to validate the URUS project, two robotic applications in an urban scenario were considered. The first application involved transporting a person or goods to a destination; and the second one was devoted to drive people slowly and orderly towards the main exit in public spaces at closing hour. In the first case, a person called, by means of a mobile phone, for a robot in order to receive the service. The robot that had the transport functionality and was closest to that person, approached



(a) (b)

Figure 3.12: Estimated x and y positions of the fire-fighter fusing all the available information. Red: Estimation provided by the decentralized approach. Green: Estimated 3σ confidence interval of the decentralized approach. Black: Centralized estimation used as ground-truth.

the person, identified the person, and guided him or her to the requested final destination. All this, with the aid of the distributed sensor network for the tasks of localization, identification, guidance, and robot navigation. In the second case, the trigger signal for the surveillance service of the public space was the closing time or a human gesture. Then, the appropriate robots available for this service approached the area where people were gathered and led them toward the exit.

The experimental sessions of the project were carried out at the Barcelona Robot Lab (see Figure 3.13), which is an outdoor urban experimental site located at the campus of the Universidad Politécnica de Cataluña (UPC), which includes 6 university buildings on a $10,000m^2$ area. This outdoor facility is also equipped with 9 WLAN antennas with complete area coverage.

The components of the URUS architecture used during the experiments described in this Chapter are shown in Figure 3.14:

- A network of 22 fixed cameras that were placed in the experimental area. The positions of several of these color video cameras are shown in Figure 3.13.
- A Wireless Sensor Network (WSN) of 30 Mica2 nodes for location purposes. These nodes were the same as in the AWARE experiments.

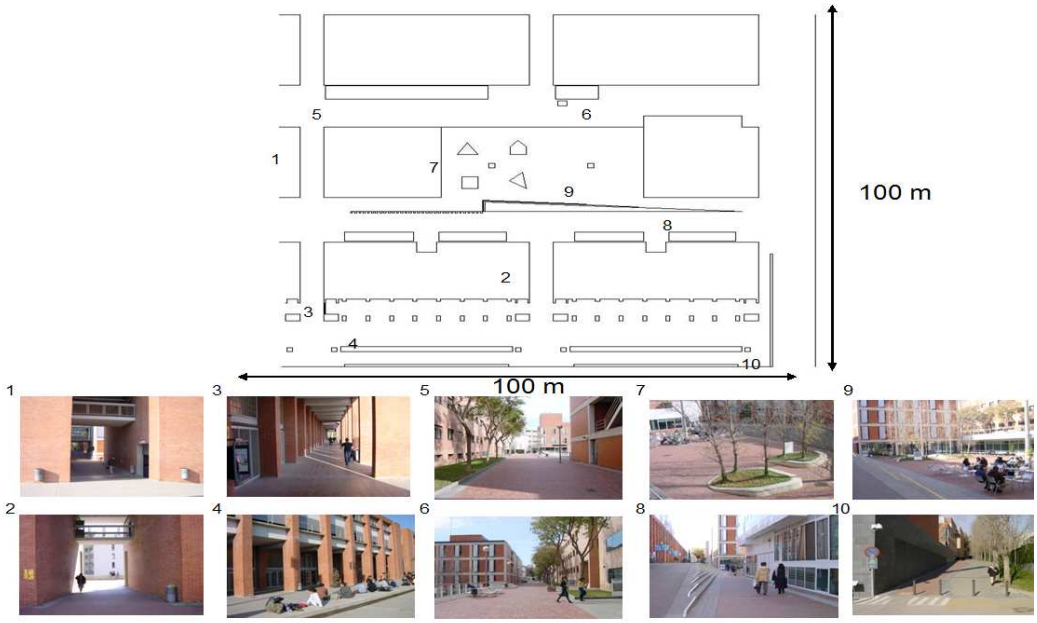


Figure 3.13: Barcelona Robot Lab. The fixed cameras offer images of the scenario from different points of view.

- Several moving ground robots equipped with cameras to detect and track people.

3.4 Real experiments in the URUS project

Cooperative tracking is one of the functionalities of the URUS project, and the algorithms presented in Chapter 2 were also applied within its framework in order to track people in an outdoor urban scenario. Moreover, one of the applications of the project was to guide a mobile robot with a person that was walking aside. For this purpose, the position of that person has to be estimated with a high accuracy by the system.

In this tracking application, the estimated state consisted of the position and velocity of the person tracked. The prediction model and the update rate for the filters were the same as in the experiments presented in this Chapter previously. In these new experiments, measurements from three different sensors are integrated: the WSN, the network of fixed cameras and the cameras on board the robots. The

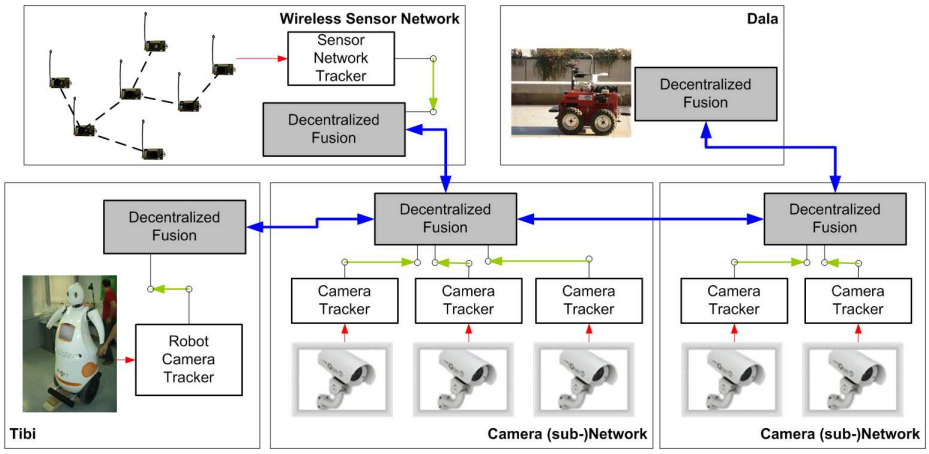


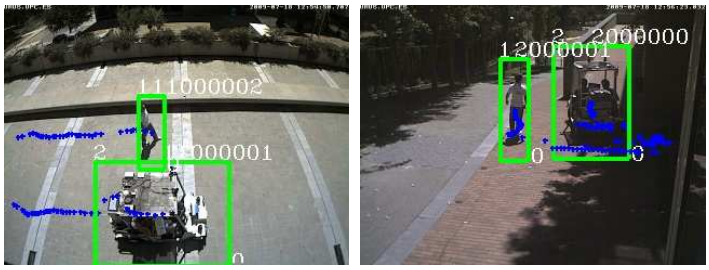
Figure 3.14: A block description of the URUS perception system. The different subsystems are integrated in a decentralized manner through a set of decentralized data fusion nodes. Locally, each system can process and integrate its data in a central way (like the WSN) or in a distributed way (like the camera network). Some systems can obtain information from the rest of the network even in the case that they do not have local sensors.

processing and models for the WSN measurements are the same as in the AWARE experiments. However, there are some differences with the algorithms of the camera measurements, which will be explained in the next Sections. Besides, it is worth mentioning that data association was also tackled as it was explained in previous Sections of this Chapter.

3.4.1 Measurements from the cameras on board the robots

The robots carried on-board cameras that were used for person tracking. These cameras could be used to obtain local estimations on the position of the person to be guided (see Figure 3.15(b)). The algorithms employed for this are based on a combination of state-of-the-art algorithms for person detection and tracking.

The person detection algorithm applied to the image is the one in (Viola and Jones, 2004). This detection module is launched when a robot is requested to guide a person who is waiting close to its location. Once the person is detected, it is tracked



(a)

(b)

Figure 3.15: (a) Tracks obtained by the camera network. (b) Tracks obtained by the camera on board the robot.

by using a tracking algorithm which is based on the CamShift technique (Bradski, 1998).

While this algorithm is able to handle temporal occlusions due to changes in illumination, the changing field of view of the camera due to the robot motion, or even the person going out of the field of view, the tracking system is not enough to maintain the track on the person continuously. Therefore, the results from the tracking and the detection applications are combined so that the robot employs the person detector whenever the tracker is lost to recover the track. The algorithm determines that the person is lost employing some heuristics, such as the track going out to the limits of the image or size restrictions on the blobs. As a result, the robots can obtain estimations of the pose of the person on the image plane.

Each camera provides a measurement composed of the position (u, v) and velocity $(\dot{u}, \dot{v})$ of the target referenced to its image plane, expressed in pixel and pixel/s, respectively. The same pin-hole model that was detailed previously in this Chapter is used here to relate these measurements to the person state. Moreover, the rotation

and position of the camera could be measured by means of the sensors on board the robots.

Finally, note that some improvements can be made in order to cope with illuminations changes (Villamizar et al., 2009). However, in general the system is not robust enough to guide a person along the whole scenario, or it could track the wrong people sometimes. Besides, from the information provided by a single camera, it is not possible to estimate the full 3D position of a person. Therefore, local camera information can be combined with information from the other sub-systems (camera network and WSN) in order to overcome these problems.

3.4.2 Measurements from the fixed cameras

There was a network of fixed cameras covering a wide area of the experimental site, which provided a foundation for the fusion of the other sensors within the system. By means of the techniques in (Gilbert et al., 2009), it is possible to track objects of interest both on and across different cameras without explicit calibration periods.

To detect moving objects within an image, the static background is modeled in a similar fashion to that originally presented by KaewTrakulPong and Bowden (2003). The foreground objects are identified from the background mask through connected component analysis. This provides a bounding box centered over each object, and a path track of each object over time is created using correlation between frames. A Kalman Filter is used to provide temporal correspondence between the detected foreground objects inter frame. A histogram is used as an object descriptor as it is spatially invariant and, through quantisation, a degree of invariance to illumination can be achieved. Each object is then given a unique label for identification. Figure 3.15(a) shows an example of tracking several moving objects on different cameras.

When the object of interest enters a new camera, the transfer of the object's label to the new camera is a challenge as the cameras may not have overlapping fields of view, making many traditional image plane calibration techniques impossible. In addition, the large number of cameras means traditional time consuming calibration is unfeasible. Therefore the approach needs to learn the relationships between the

cameras automatically. This is achieved by way of two individually weak cues, modeling the color, and movement of objects inter camera. These two weak cues are then fused to determine if objects have been previously tracked on another camera or are new object instances. The approach learns these camera relationships, though unlike previous work does not require *a priori* calibration or explicit training periods. By incrementally learning the cues over time, the accuracy is able to increase without any supervised input.

With these previous techniques, the cameras can provide measurements of the position of the object on the image plane in pixel coordinates (u, v). However, since the all the cameras were fixed, a prior calibration process was perform in order to obtain homographies for all of them. Thus, the measurements on the image plane can be transformed into measurements referred to a 3D coordinate system (3.8). After this transformation, the state to estimate (person's position) and the measurement are expressed in the same 3D coordinates, so their relation is straightforward.

$$z_t^{cam} = \begin{pmatrix} x_t^{cam} & y_t^{cam} & z_t^{cam} \end{pmatrix}^T = H_{cam} \begin{pmatrix} u_t & v_t \end{pmatrix}^T \tag{3.8}$$

3.4.3 Experiment 1

In this first experiment, one moving robot, a WSN of 30 nodes and 7 fixed cameras were used. During the experiment, one person was following the robot, which was manually controlled. In this case, there were a PSS on the robot, one for the WSN and 2 for the fixed cameras, one integrating measurements from 3 cameras and the other from 4 cameras.

Along the whole trajectory of the person, of more than 350 meters, there were gaps in the camera coverage. Moreover, the robot lost its track in certain moments due to the changes in illumination, etc. Finally, the WSN coverage was limited to a certain part of the campus.

The estimated position of the person with the full system running is shown in Figure 3.16(a). The total length of the experiment is around 350 meters and 5 minutes. The person is usually besides the robot (which means that the x or y coordinates are

similar). The system is able to maintain the estimation on the person position for the full trajectory. There is WSN coverage between 0 and 150 seconds, approximately. A specific interval of the trajectory is zoomed in Figure 3.16(b). In this part, only information from the WSN and the robot are available. Although the WSN measurements have low precision, they enable the filter to bound the error from the monocular camera. At time 75 seconds approximately, the person enters under coverage of the camera network, which leads to a remarkable reduction in uncertainty.

During the experiments, the communication between the PSSs on board the robots and the ones related to the camera network and the WSN was done using Wi-Fi and 3G technologies. A software running on the robot was able to measure the quality of the Wi-Fi link, and switch to 3G whenever this quality dropped below a certain threshold.

The switch between communication networks created from time to time blind periods of several seconds. Moreover, although 3G had more stable coverage in the scenario, it had also lower bandwidth and higher latencies. In order to cope with this, it was very important the use of a decentralized system with delayed states. The use of decentralized nodes enabled the system to cope with communication drops, as in the mean time, the local nodes were accumulating information. When the communication links were recovered, the nodes exchanged their estimations. Moreover, as delayed states were considered, this delayed information (and also information delayed due to the latencies) could be fused in a correct way, and no information was lost.

3.4.4 Experiment 2

In this second experiment, the decentralized estimation is compared to a centralized implementation. The results are shown in Figure 3.17. In this experiment, 4 different cameras and the WSN were running. For the centralized implementation, all the information was received and fused in a single PSS; whereas for the decentralized case three PSSs were used (two of them processing information locally from 2 cameras; the remaining one processing information locally from the WSN). The estimation obtained by one of the PSS is shown in Figure 3.17. It can be seen how, with some

latencies depending on the conditions, a decentralized node obtains an estimation quite close to the centralized one, even though some information is lost due to the conservative fusion rule employed in this case (Covariance Intersection, see Chapter 2). Again, as in the AWARE setup, the flexibility required in the URUS network did not allow the enforcement of tree-like topologies.

3.5 Conclusions

This Chapter presents some results obtained during real experiments for a cooperative tracking application. The algorithms described in Chapter 2 are particularized for such an application and tested in two different scenarios: a disaster management scenario and an urban scenario. All the experiments included in the Chapter were performed within the framework of the European projects AWARE and URUS, whose experimental setups are also described.

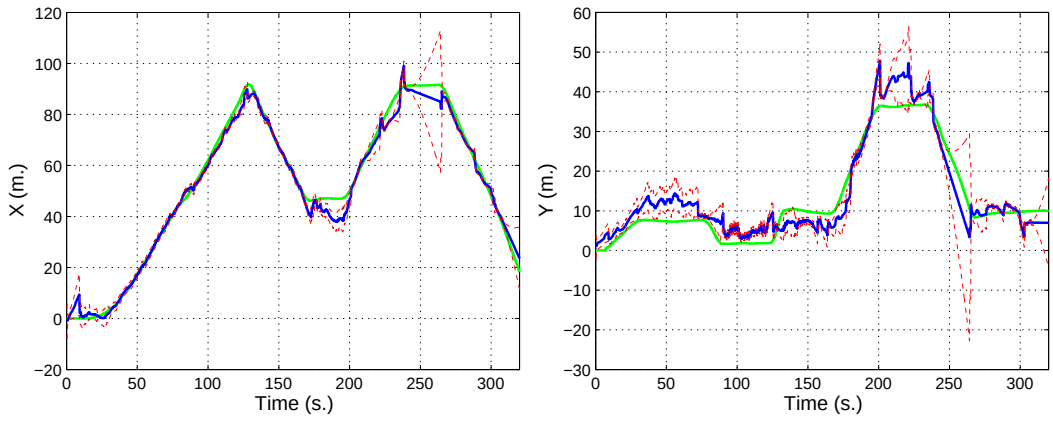
The various sensors used and their corresponding models are explained as well as part of the real-time implementation of the whole system. Due to the nature of some of the heterogeneous sensors, non-linear models are considered. Besides, no constraints about the network topology are taken into account and conservative data fusion is done. Even though the information is fused in a conservative manner, the use of delayed states led to more robust estimations, thus enabling the system to cope with temporal communication breakdowns efficiently.

In general, the experimental results showed that the proposed approach is able to track the position of a moving object in a fully decentralized manner with small errors with respect to a centralized filter, obtaining results with similar mean (an error of around one meter) and standard deviation (a difference of around half a meter).

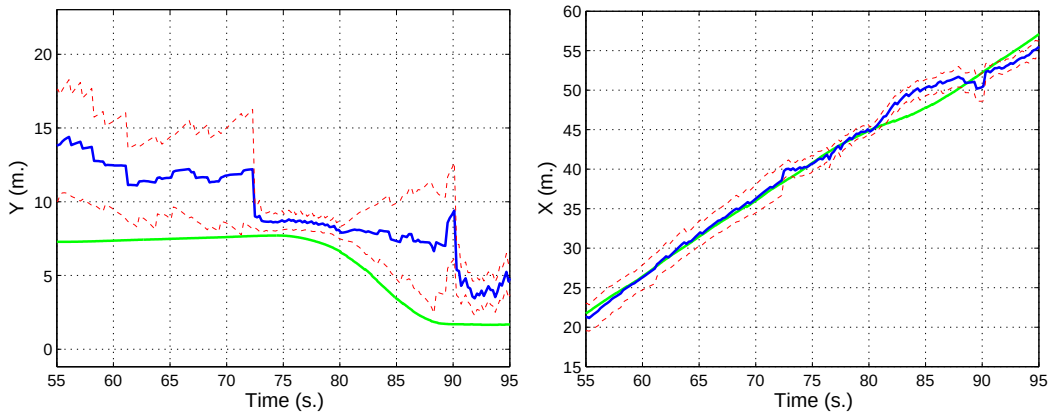
The combination of information provided by heterogeneous sub-systems achieves accurate tracking in more challenging situations. Actually, very complex algorithms employing just one source of information are usually unable to cope with all the potential situations in real scenarios, which are affected by changes in illumination and clutter and in which a wide area must be covered. Therefore, the combination

of complementary systems can be useful for these applications. For instance, signal-based localization is less accurate than camera-based localization, but the former is less affected by occlusions. Cameras attached to moving robots usually have narrower fields of view, but they can adapt to cover places occluded for a camera network. Instead, camera networks can provide overall information on the scene.

Finally, the decentralized scheme proposed solved the scalability issues, since the addition of new sub-systems does not affect the rest of the perception system in terms of storage, as only local communication and local processing are used.



(a)



(b)

Figure 3.16: Estimated position of the person (blue) compared to the position of the robot (green). Dashed lines represent the standard deviation of the estimation. (a) Complete trajectory. (b) A section of the trajectory. The person is following the robot with the same x coordinate up to time 80 seconds. Then the robot changes orientation. The person is separated from the robot around 3-4 meters.

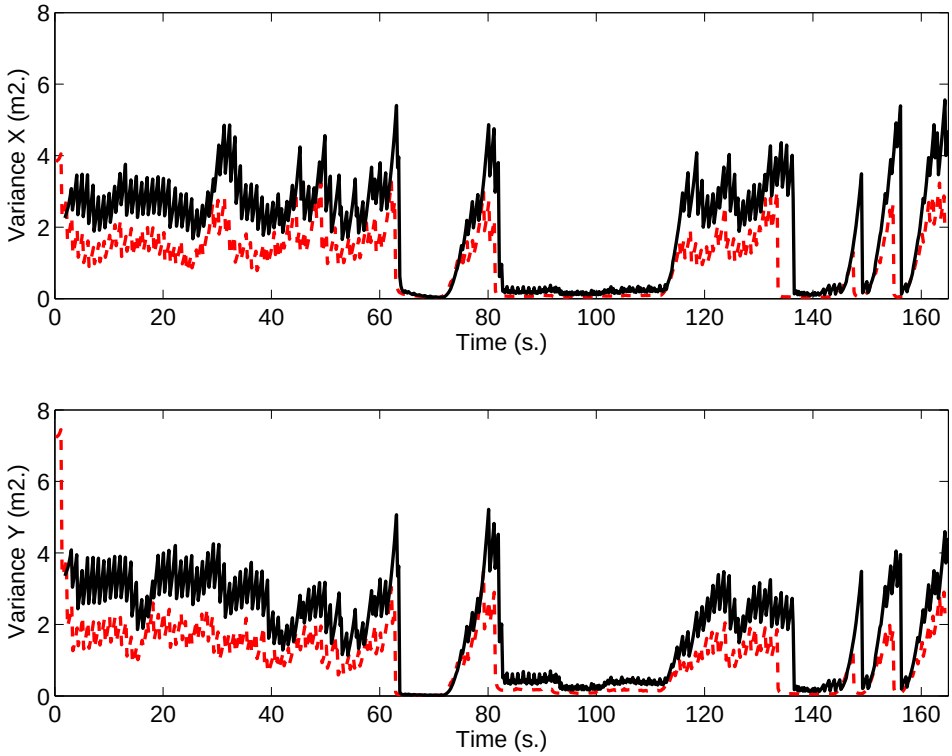


Figure 3.17: Estimated variance by a central node receiving all the information (dashed and red) compared to the estimation in a decentralized node (solid and black).

Chapter 4

Partially observable Markov decision processes for active perception

In Chapter 2, a probabilistic framework was presented in order to estimate the state of the environment in a decentralized manner with a group of robots. In Chapter 3, such framework was applied to cooperative tracking and extensive experiments with different networks of robots and sensors were shown. Basically, these previous Chapters presented a solution to the problem of cooperative perception with a team of robots. However, the final goal of this Thesis is cooperative active perception, where the robots in a team need to decide which actions are better to improve their perception. For this purpose, unlike those in Chapters 2 and 3, the approach must incorporate the ability of making decisions into the system. Partially Observable Markov Decision Processes (POMDPs) extend the approach shown in Chapters 2 and 3 and create an active perception framework that can cope with uncertain environments. Thus, not only would robots perceive their environment, but could also make appropriate decisions to enhance that information.

Therefore, this Chapter presents POMDPs as a solution for active perception and explains their mathematical description. POMDPs (Thrun et al., 2005; Kaelbling

et al., 1998) are a rich and general mathematical framework that facilitate the modeling of stochastic environments for taking optimal decisions. Due to their well-founded theory and their elegant design when modeling dynamic systems, they have been widely applied by the artificial intelligence and robotics communities to planning under uncertainties. POMDPs extend the Markov Decision Processes (MDPs) (Thrun et al., 2005) by considering that the system's information is not fully observable (it cannot be accessed directly but through noisy sensors). Moreover, the mathematical framework that POMDPs offer is quite general, since varied optimization objectives can be considered just by designing properly a function that they optimize. Furthermore, the models described by POMDPs can also be learnt online, which increases the number of applications for which they are suitable.

Even though there are many algorithms that can solve these kinds of models for large belief spaces, the majority of them work with discrete state spaces (the possible states of the system are finite). Due to the lack of efficient algorithms for solving POMDPs in continuous spaces, this Thesis focuses on discrete POMDPs. Besides, as it will be shown, there are not so many algorithms that can cope with multi-agent POMDPs (i.e. several agents or robots taking decisions) when the state space is moderately large. Therefore, the Thesis intends to contribute by proposing alternative approaches that enable the use of current POMDP techniques in multi-robot systems.

This Chapter describes the mathematical models for single-agent POMDPs (applied to active perception) and some of the more extended algorithms that solve these models. Besides, the Chapter surveys the literature to give an overview of the different types of POMDPs and their applications. Finally, existing POMDP-based models for multi-agent systems are introduced and discussed at the end of the Chapter.

4.1 POMDP framework

4.1.1 Mathematical model

Formally, a POMDP is defined for a single agent (or robot) by the tuple $\langle S, A, Z, T, O, R, h, \gamma \rangle$. The meaning of each component is the following:

State space

All the relevant information about the environment is encoded in the state, which is not fully observable. This means that this state cannot be observed directly, and a probability distribution over the set of possible states has to be maintained based on observations from noisy sensors. The system's state at each time step is defined by $s_t \in S$, where S is the finite set of all the possible states.

Action space

The agent in the environment can take an action each time step. This actions can modify the state in a stochastic manner. A is the finite set of possible actions, whereas $a_t \in A$ is the action taken at a certain time step.

Observation space

Given a time step t, after executing an action, an agent can make a measurement or observation $z_t \in Z$, where Z is the finite set of all the possible observations. An observation is information about the environment that can be perceived by the agent.

Transition function

When an agent executes an action, the state can vary probabilistically. This probability density function is modelled by the transition function $T : S \times A \times S \longrightarrow [0, 1]$, where $T(s', a, s) = p(s_t = s' | a_{t-1} = a, s_{t-1} = s)$. It represents the probability of ending at state s' if the agent performs the action a at state s.

Observation function

The observations gather information from the current state and are related probabilistically to this state. This probability density function is modelled by the observation function $O : Z \times A \times S \longrightarrow [0, 1]$, where $O(z, a, s') = p(z_t = z|a_{t-1} = a, s_t = s')$. It gives the probability of observing z if the action a is performed and the resulting state is s'.

Reward function

A POMDP selects the best actions so that a utility function is optimized. Thus, the behavior of the system is determined by this function, which will depend on a reward function. $R : S \times A \longrightarrow \Re$ is that reward function, where $R(s, a)$ is the reward obtained by executing action a at state s. In fact, the reward itself may include a cost associated with the execution of the given action. Here, the reward is assumed to be bounded. Since it can model quite complex goals, the reward function is a very powerful tool, and hence, its correct design is crucial.

Horizon and discount factor

Then, the goal of an agent is to maximize the expected reward earned over some time frame. The horizon h defines this time frame by specifying the number of time steps the agent can plan for. Thus, the objective is to maximize the sum:

$$E\left[\sum_{t=0}^{h} \gamma^t r_t\right] \tag{4.1}$$

where r_t is the reward at time t, $E[\ ]$ is the mathematical expectation (with respect to the possible states), and $\gamma \in [0, 1)$ is a discount factor, which ensures that the sum in (4.1) is finite when $h \to \infty$. Moreover, this factor indicates how rewards should be weighted at different time steps.

4.1.2 Belief computation

Given that the state is not directly observable, the actual state cannot be known by an agent. Instead, the information about the environment should be encoded in a *belief distribution*, which indicates a probably density function over the state space.

In order to calculate the current belief state $b(s_t)$ from a initial belief state $b(s_0)$, a complete trace of all the observations made and actions executed until t would be necessary. This trace is called the *history* of the system and grows as time goes on. However, due to the Markov assumption, it is well-known that the belief state at a certain time step can be calculated by just considering the belief at the previous step:

$$b(s_t) = p(s_t|z_t, a_{t-1}, z_{t-1}, \cdots, a_0, b(s_0)) = p(s_t|z_t, a_{t-1}, b(s_{t-1})) \tag{4.2}$$

Since the belief distribution $b(s_t)$ is a sufficient statistic for the history, instead of maintaining a growing sequence of past observations and actions, this belief can be updated recursively by using only the previous belief $b(s_{t-1})$, the most recent action a_{t-1} and the most recent observation z_t. Thus, a belief $b(s)$ can be updated assuming an action a and an observation z with a Bayesian filter in the following way:

$$b_a^z(s') = \frac{1}{p(z|a,b)} O(z, a, s') \sum_{s \in S} T(s', a, s) b(s) \tag{4.3}$$

where $p(z|a, b)$, the probability of observing z after taking action a in belief b, acts as a normalizing constant such that b_a^z remains a probability distribution:

$$p(z|a,b) = \sum_{s' \in S} O(z, a, s') \sum_{s \in S} T(s', a, s) b(s) \tag{4.4}$$

Applying the definitions of T and O, it can be seen that (4.3) is equivalent to Equation 2.3 in Chapter 2. The only differences are that (2.3) expressed the belief update for continuous state spaces and assumed that there were no actions from the agents that could varied the state.

4.1.3 Optimal policy

Once the agent has computed the belief state, the next and most relevant step is to choose which is the best action for that given belief. This action is determined by a strategy or policy $\pi(b)$, which defines the action to execute for all possible beliefs the agent may encounter. Therefore, the policy is an attempt to map beliefs to actions so that the amount of reward earned over the time horizon is maximized. The main objective of a POMDP algorithm is to find this policy in the form:

$$\pi(b) \longrightarrow a \tag{4.5}$$

where b is a belief distribution and a is the action chosen by the policy π.

The policy $\pi(b)$ is a function over the continuous set of belief states, and it can be characterized by a value function $V^{\pi}(b)$, which is defined as the expected future discounted reward that the agent can gather by following π starting from belief b:

$$V^{\pi}(b) = E\left[\sum_{t=0}^{h} \gamma^t r(b_t, \pi(b_t)) | b_0 = b\right] \tag{4.6}$$

where $r(b_t, \pi(b_t)) = \sum_{s \in S} R(s, \pi(b_t)) b_t(s)$. Therefore, the optimal policy π^* is the one that maximizes that value function V^{π}:

$$\pi^*(b) = \arg\max_{\pi} \; V^{\pi}(b) \tag{4.7}$$

Note that computing an optimal policy for a POMDP may be challenging mainly for two reasons. The first one is the dimensionality of the belief space. Even for a finite set of $|S|$ states, π is defined over an $(|S| - 1)$-dimensional continuous belief space, which can be quite complex depending on the problem. The second source of complexity is the length of the history. A POMDP can be solved by searching through the space of possible histories, but the number of distinct possible action-observation histories grows exponentially with the planning horizon. An algorithm to solve POMDPs should tackle both sources of complexity in order to be efficient.

4.1.4 POMDP model for multiple robots

The previous framework can be extended to multi-agent systems in a straightforward manner. Provided that a set of N robots is considered, the Multi-agent POMDP (MPOMDP) model can be used, which expands the single-agent model as follows:

- Each robot i can execute an action a^i from a finite set A_i. The set of joint actions is $a^J \in A_1 \times \cdots \times A_N$.
- Each robot i can receive an observation z^i from a finite set Z_i. The set of joint observations is $z^J \in Z_1 \times \cdots \times Z_N$.
- The transition function $T(s', a^J, s)$ is defined over the set of joint actions.
- The observation function $O(z^J, a^J, s')$ relates the state to the joint action and the joint observation.
- The reward function is now defined over the joint set of states and actions $R : S \times A_1 \times \cdots \times A_N \to \mathbb{R}$. That common reward includes all the individual objectives for each agent as well as the team's ones.

Given the above definitions, the goal in this case is to compute an optimal joint policy $\pi^* = \{\pi_1, \cdots, \pi_N\}$ that maximizes the expected discounted reward. That policy is defined over the joint belief, which is the belief obtained with all the information of the team. This belief must be updated considering joint actions and observations, and the transition and observation functions defined above.

This Thesis is aimed at active perception with multi-robot teams, so this POMDP model and its derivatives will be used for such a purpose. The model itself can cope with the main problems in the Thesis: it considers systems with multiple robots; it allows cooperative perception; and optimal decision-making under uncertain environments. Moreover, the goals of the system are encoded into its reward function, which designed properly, will lead it to active perception.

In general, within the POMDP framework, there are a lot of possible approaches to perform active perception. Most of them consist of rewarding preferentially actions

that gather more information from the environment. For instance, sensors may be focused on areas where the lack of information is higher, or robots should be sent to certain areas according to the characteristics of their sensors. In particular, in the cooperative tracking application presented in Chapter 3, the perception is improved by reducing the uncertainty in the target position. Therefore, configurations that achieve this goal should be given a higher reward. For instance, cameras that point at the target from different directions, or robots that explore areas with a higher uncertainty.

4.2 POMDP solvers

4.2.1 Value iteration

A key result for POMDPs is that the value function at horizon n can be constructed from the value function at horizon $n-1$ by means of the Bellman Equation (Thrun et al., 2005):

$$V_n(b) = \max_a \left[\sum_{s \in S} R(s,a) b(s) + \gamma \sum_{z \in Z} p(z|a,b) V_{n-1}(b_a^z) \right] \tag{4.8}$$

where b_a^z is the belief update in (4.3). Thus, given an initial value function V_1, the value function at a desired horizon h can be obtained recursively by iterating (4.8). This procedure is called *value iteration* (Sondik, 1971) and is the base of many POMDP solvers.

Another essential result is shown in (Smallwood and Sondik, 1973), where the optimal value function for any finite horizon is proved to be piecewise linear and convex (PWLC). It means that the value function at iteration n can be expressed by a set of vectors: $\Gamma_n = \{\alpha_1, \alpha_2, \cdots, \alpha_{|V_n|}\}$. Each of these vectors, so-called α-vectors, is an $|S|$-dimensional hyperplane and defines a region in the belief space for which this vector is the maximizing element of V_n:

$$V_n(b) = \max_{\alpha \in \Gamma_n} \sum_{s \in S} \alpha(s) b(s) \tag{4.9}$$

Besides, there is an action $a \in A$ associated with each α-vector. That would be the optimal action to execute at the current step in case its associated vector is the one that maximizes V_n. In other words, the value of V_n in a certain belief state is the maximum value returned by one of the α-vectors for this belief state, whereas the best action is the one associated with the α-vector that returned this best value.

There are many exact algorithms in the literature exploiting the PWLC properties of the value function (Sondik, 1971; Smallwood and Sondik, 1973; Monahan, 1982; Littman, 1996; Cassandra et al., 1997; Zhang and Zhang, 2001). However, the main problem that these approaches encounter is that the number of α-vectors required to represent the value function grows exponentially with the number of observations at each iteration. Hence, the computational complexity at each iteration also grows, so the set Γ_n should be efficiently pruned in order to optimize the performance.

4.2.2 Perseus

Apart from the growing size of the value function set explained above, another major source of intractability of exact POMDP solutions is the fact that an optimal action must be computed for every possible belief point within the belief space. This can be tackled by computing an approximate solution in which only a finite set of belief points is considered. Thus, (4.8) would only be computed a bounded number of times at every iteration, reducing the computational cost and the number of α-vectors generated.

Methods using this approximation are called *point-based* methods and they reach a compromise between the loss of optimality and the ability to compute policies for larger problems. In particular, Perseus (Spaan and Vlassis, 2005), one of the most well-known algorithms performing point-based value iteration, is described thoroughly in this Section.

First, the inner product $(\cdot)$ is introduced to simplify the notation as $\beta(s) \cdot \gamma(s) = \sum_{s \in S} \beta(s)\gamma(s)$. Using this operator, (4.9) can be rewritten as:

$$V_n(b) = \max_{\alpha \in \Gamma_n} \alpha \cdot b, \tag{4.10}$$

whereas (4.8) is transformed as:

$$V_n(b) = \max_a \left[R_a \cdot b + \gamma \sum_{z \in Z} p(z|a,b) V_{n-1}(b_a^z) \right], \tag{4.11}$$

where $R_a(s) = R(s,a)$.

When performing value iteration, applying (4.11) for a particular belief point is straightforward. Thus, given V_{n-1} and a certain belief point b, the vector $\alpha^b \in \Gamma_n$ such that:

$$\alpha^b = \arg\max_{\alpha \in \Gamma_n} \; b \cdot \alpha \tag{4.12}$$

can be calculated easily. This operation is denoted by $\alpha^b = \texttt{backup}(b)$ and it computes the optimal vector for a given belief b by back-projecting all the vectors in the current horizon value function one step backward from the future and returning the vector that maximizes the value of b.

Then, Equation (4.11) can be expanded:

$$V_n(b) = \max_a \left[R_a \cdot b + \gamma \sum_{z \in Z} p(z|a,b) V_{n-1}(b_a^z) \right] \tag{4.13}$$

$$= \max_a \left[R_a \cdot b + \gamma \sum_{z \in Z} p(z|a,b) \max_{\alpha \in \Gamma_{n-1}} \sum_{s' \in S} b_a^z(s') \alpha(s') \right] \tag{4.14}$$

$$= \max_a \left[R_a \cdot b + \gamma \sum_{z \in Z} \max_{\alpha \in \Gamma_{n-1}} \sum_{s' \in S} \sum_{s \in S} p(z|a,s') p(s'|a,s) b(s) \alpha(s') \right] \tag{4.15}$$

$$= \max_a \left[R_a \cdot b + \gamma \sum_{z \in Z} \max_{\alpha \in \Gamma_{n-1}} b \cdot \left(\sum_{s' \in S} p(z|a,s') p(s'|a,s) \alpha(s') \right) \right] \tag{4.16}$$

Therefore, the *backup* operator can be defined as:

$$\texttt{backup}(b) = \arg\max_{a \in A} \; b \cdot \alpha_a, \text{ where} \tag{4.17}$$

$$\alpha_a = R_a + \gamma \sum_{z \in Z} \underset{\alpha \in \Gamma_{n-1}}{\arg\max}\ b \cdot \left(\sum_{s' \in S} p(z|a,s') p(s'|a,s) \alpha(s') \right) \tag{4.18}$$

As said before, even though the backup operation for the whole belief space may be extremely hard, such a complexity is alleviated when operating on a finite set of points B. Perseus, like other point-based algorithms, relies on the assumption that the policy generated will also work properly for belief points not included in B. The fulfillment of that assumption depends on the nature of the POMDP problem itself, and of course it is highly related to the way in which the belief points are selected. In many cases, a good selection of B can be sufficient to calculate a policy that performs well for the whole belief space.

Therefore, an essential issue for the performance of the policy is how to build the set B. A first approach may be to choose the belief points in a regular grid (Lovejoy, 1991). A second simpler alternative is to sample randomly a number of beliefs. Finally, there is another option consisting of simulating the POMDP model with some initial policy and gathering visited beliefs (Pineau et al., 2003, 2006). In this case, B will be composed of beliefs which are more likely to appear in real situations.

An additional key idea in Perseus is that, at each iteration, the value function is improved (or at least it is not decreased) for all the belief points in B by performing the backup operation on a subset of the points. Thus, the value function is always increasing its value at each iteration for the whole set B:

$$V_{n-1}(b) \leq V_n(b),\ \forall b \in B \tag{4.19}$$

The complete procedure for Perseus is summarized in Algorithm 3. First, a set of beliefs B is built by means of any of the techniques previously mentioned. The value function is initialized (line 2) as a single vector with all its components equal to the minimum possible reward that the agent may obtain during the execution of any policy. Thus, the initial value function is assured to be a lower bound on the optimal

Algorithm 3 Perseus$(S, A, Z, T, O, R, h, \gamma)$

1: Sample set of reachable beliefs B
2: Initialize $\Gamma_0 = \{\frac{1}{1-\gamma} \min_{s,a} R(s,a)\}$
3: $n = 0$
4: **repeat**
5: $\quad n = n + 1$
6: $\quad \Gamma_n = \emptyset$
7: $\quad \tilde{B} = B$
8: $\quad$ **while** $\tilde{B} \neq \emptyset$ **do**
9: $\quad\quad$ Sample $b \in \tilde{B}$
10: $\quad\quad \alpha^* = \texttt{backup}(b)$
11: $\quad\quad$ **if** $b \cdot \alpha^* \geq V_{n-1}(b)$ **then**
12: $\quad\quad\quad \Gamma_n = \Gamma_n \cup \{\alpha^*\}$
13: $\quad\quad$ **else**
14: $\quad\quad\quad \alpha' = \arg\max_{\alpha \in \Gamma_{n-1}} b \cdot \alpha$
15: $\quad\quad\quad \Gamma_n = \Gamma_n \cup \{\alpha'\}$
16: $\quad\quad$ **end if**
17: $\quad\quad \tilde{B} = \{b \in \tilde{B} : V_n(b) < V_{n-1}(b)\}$
18: $\quad$ **end while**
19: **until** convergence
20: **return** Γ_n

value function. Then, Perseus performs a number of value iteration steps approximating new value functions. The Algorithm can stop at a fixed number of iterations or when some convergence criterion is met (line 19). Even though several criteria are possible, the most common is to measure the improvement between consecutive estimations $\max_{b \in B}(V_n(b) - V_{n-1}(b))$. Another possibility would be to measure the changes in the policies, i.e. the number of $b \in B$ which vary their optimal action from one step to the next one.

At each value iteration, Perseus improves $V_{n-1}(b)$ for all $b \in B$ by backing up as few belief points as possible. The auxiliary set $\tilde{B}$ keeps the non-improved points for the current iteration. The Algorithm proceeds sampling belief points $b \in \tilde{B}$ and applying the backup operator to them (lines 9 and 10). If the value for b is improved, the new generated α-vector is added to the value function $V_n(b)$. Otherwise, the α-vector from Γ_{n-1} which maximized the value for that b is maintained in Γ_n (lines

11 to 16). Every time a new vector is created and included in Γ_n, all the improved belief points are removed from $\tilde{B}$ because no additional backup is necessary for them (line 17). The idea is that performing the backup for a subset of B and generating a reduced number of α-vectors, all the belief points in B could be improved, mainly at the first iterations. Once the new values for all points in B are greater or equal to the former ones, $\tilde{B}$ is empty and the current iteration is over (line 8).

Given the increasing nature of the value function computed by Perseus, the convergence proof is simple. Since the value function is initialized to the minimum possible value and is growing at every step, it will always be below the optimal $V^*(b)$. Besides, the number of vectors required for this convergence is not as large as for other methods. Due to the fact that a backup vector for a belief also improves other beliefs, fewer vectors are needed to build the complete value function.

Finally, note that Perseus loses somehow the notion of exact planning horizon. Since the belief points are backed up asynchronously to focus on the regions of interest, performing n iterations will not result exactly in a plan n steps into the future.

4.2.3 Symbolic Perseus

Point-based algorithms produce near-optimal value functions by means of policies generated with a reduced number of α-vectors. Despite this low complexity, these policies can be good enough and solve large POMDPs. Nevertheless, due to the possible high dimensionality of each α-vector, these algorithms are still restricted to work with problems of thousands states.

Factored POMDPs (Poupart, 2005) are based on the fact that belief states are sparse and they usually maintain more information than necessary. Provided that there exists a significant structure (e.g., conditional independence, context-specific independence, additive separability), the dimensionality of vectors can be reduced by using compact representations, such as algebraic decision diagrams (ADDs) (Hoey et al., 1999).

In a POMDP, the reward, transition and observation functions are specified over the whole state space S. However, sometimes the state can be represented by a set

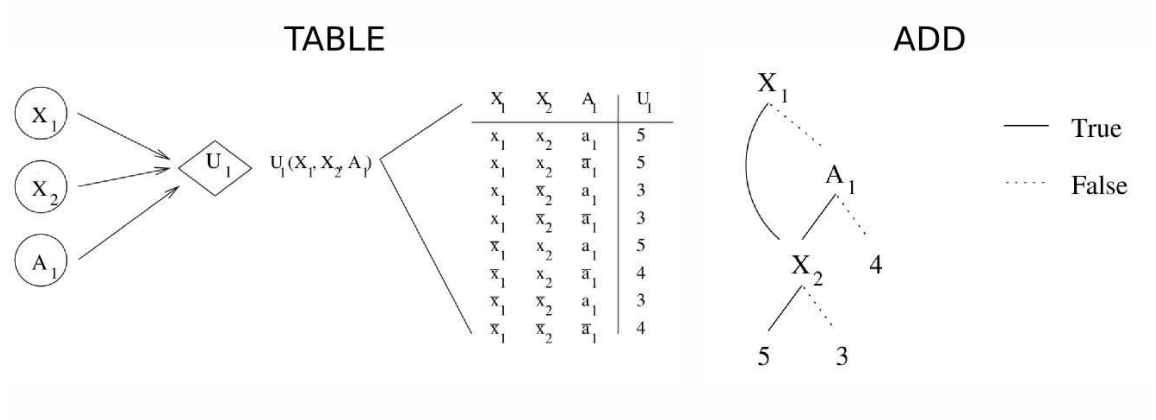


Figure 4.1: Function U_1 can be represented by a table (left) and an ADD (right).

of random variables $\{\chi_i\},\ i = 1, \cdots, k$ with domains $\{S_i\},\ i = 1, \cdots, k$, such that $S = S_1 \times S_2 \times \cdots \times S_k$. The same can be applied to actions and observations in order to decompose them into sets of variables. Thus, in many domains a factored representation of the reward, transition and observation functions in terms of these variables instead of the complete states, actions or observations is feasible.

Factored representations exploit the *conditional independence*, i.e. the fact that some variables are probabilistically independent of each other when the values of other variables are held fixed. In this case, the conditional probabilities $T(s', a, s)$ and $O(z, a, s')$ can be expressed using the well-known dynamic Bayesian networks (DBNs) (Dean and Kanazawa, 1989) and their conditional probability tables (CPT) (Poupart, 2005). Actually, they can even be further compressed by using Algebraic Decision Diagrams (ADDs).

An ADD is a tree that stores in its leaves the values of a function defined over some variables. Moreover, its branches are allowed to merge together. Thus, a function defined over a set of variables can be represented compactly by aggregating together values that are identical. Figure 4.1 shows an example of a same function represented by a table and an ADD. It can be seen how the decision tree can encode with only 3 leaves the 8 values of the table.

In (Poupart, 2005), an extension of Perseus is presented. The aim is to overcome the explained curse of dimensionality in the performance of Perseus by combining a

point-based method with ADDs, and thus, be able to solve efficiently larger POMDPs. The resulting approach is called *Symbolic* Perseus, since it is a version of Perseus where matrix operations are carried out symbolically. Hoey et al. (1999) describe how to implement efficiently classic value iteration with ADDs. That integration can be applied to Perseus directly. The idea is simply to represent α-vectors and beliefs by ADDs and perform all the matrix operations and backups symbolically. Reward functions and CPTs for the transition and observation models are also represented by ADDs. Since ADDs aggregate together the entries of vectors that are identical, matrix operations can be done efficiently by performing the arithmetic operations applied to identical entries only once.

Some further steps are taken in Symbolic Perseus in order to optimize the process. Depending on the problem, it may be the case where the generated beliefs and α-vectors have not got many identical but similar values. In these cases, the compact representation introduced by the ADDs is no longer helpful, leading to large trees. Therefore, in Symbolic Perseus non-equal branches of the ADDs are aggregated as long as the absolute difference between the values of their leaves is below a certain threshold (St-Aubin et al., 2001). Besides, the number of α-vectors for each value function is bounded, so the backup process within a value iteration is stopped when this limit is reached. The last approximation consists of factorizing the belief states as the product of their independent marginal probabilities.

Symbolic Perseus (see Algorithm 4) is quite similar to the original Perseus, but ADDs and factored representations are introduced as data structures. Nonetheless, there are a couple of variations with regard to the version in Algorithm 3 that should be noticed. First, the value function in Symbolic Perseus is not always increasing for all $b \in B$. For a sampled b, the backed-up vector α^* is calculated (line 11), and this vector is added to the new value function V_n if it improves at least one of the values for the current V_n (lines 12 to 14). Thus, it may occur that α^* is added anyway without improving the particular point b, i.e. $V_n(b) < V_{n-1}(b)$. Furthermore, now the value function can converge to a near-optimal solution oscillating up and down, so the initial value function is no longer required to be a lower bound. Here, the initial value function is set to the immediate reward obtained for every action instead of the

Algorithm 4 Symbolic Perseus$(S, A, Z, T, O, R, h, \gamma)$

1: Sample set of reachable beliefs B
2: Initialize $\Gamma_0 = \{R_a | a \in A\}$
3: $n = 0$
4: **repeat**
5: $n = n + 1$
6: $\Gamma_n = \emptyset$
7: $\tilde{B} = B$
8: **while** $\tilde{B} \neq \emptyset$ **and** $|\Gamma_n| < MAXSIZE$ **do**
9: Sample $b \in \tilde{B}$
10: $\tilde{B} = \tilde{B} \setminus \{b\}$
11: $\alpha^* = \texttt{backup}(b)$
12: **if** $\Gamma_n = \emptyset$ **or** $\max_{b \in B}(b \cdot \alpha^* - V_n(b)) > 0$ **then**
13: $\Gamma_n = \Gamma_n \cup \{\alpha^*\}$
14: **end if**
15: $\tilde{B} = \{b \in \tilde{B} : V_n(b) < V_{n-1}(b)\}$
16: **end while**
17: **until** convergence
18: **return** Γ_n

minimum possible reward (line 2). Finally, note that each value iteration is stopped when the size of Γ_n reaches its maximum allowed value ($MAXSIZE$) (line 8).

4.2.4 Sarsop

Sarsop, which stands for Successive Approximations of the Reachable Space under Optimal Policies, is another point-based algorithm that exploits the notion of optimally reachable belief spaces to improve computational efficiency. For further details (Kurniawati et al., 2008) can be consulted.

Many point-based algorithms sample B, a subset of belief points reachable from a given initial point $b_0 \in B$, and keep it fixed during the execution of the algorithm. That is the case of Perseus, for instance. However, Sarsop attempts to sample a subset of belief points reachable from b_0 under optimal sequences of actions B^*, which would be smaller than an arbitrary B. Since the optimal sequences of actions constitute

Algorithm 5 Sarsop($S, A, Z, T, O, R, h, \gamma$)

Initialize Γ_0, representing the lower bound $\underline{V}$. Initialize the upper bound $\overline{V}$.
Insert the initial belief point b_0 as the root of the tree $\mathcal{T}_R$
$n = 0$
repeat
  `sample`($\mathcal{T}_R$,Γ_n)
  Choose a subset of nodes from $\mathcal{T}_R$. For each chosen node b, `backup`($\mathcal{T}_R$,Γ_n,b)
  `prune`($\mathcal{T}_R$,Γ_n)
  $n = n + 1$
until Gap between $\underline{V}$ and $\overline{V}$ is small enough or time limit is reached.
return Γ_n

exactly the POMDP solution, which is unknown in advance, this optimal subset must be approximated.

The key idea in Sarsop is to compute successive approximations of B^* converging iteratively. A heuristic exploration is performed to sample the belief space by means of an online learning technique. Besides, a bounding technique to avoid sampling in regions that are unlikely to be optimal is used. This sampling method leads to a substantial gain in computational efficiency.

Sarsop can be summarized in Algorithm 5 and is mainly based on three functions, `sample`, `backup` and `prune`. The set of sampled beliefs is stored in a tree $\mathcal{T}_R$, where each node represents a sampled point and the root is b_0. Moreover, in order to focus the sampling on B^*, Sarsop maintains both a lower bound $\underline{V}$ and an upper bound $\overline{V}$ on the optimal value function V^*. The set of α-vectors Γ can represent a lower bound with a correct initialization (e.g. using a fixed-action policy). The upper bound is obtained by means of a sawtooth approximation (Hauskrecht, 2000) and it can be initialized by using the MDP or the Fast Informed Bound technique.

The `sample` function is in charge of sample new beliefs at each iteration and include them into the tree. In order to sample a new point b', a node b is picked up from $\mathcal{T}_R$ and an action and an observation are chosen according to suitable probability distributions or heuristics. Then, b' can be computed with (4.3) and inserted into $\mathcal{T}_R$ as a child of b. Thus, all the sampled beliefs are reachable from b_0. Moreover, Sarsop uses the upper and the lower bounds to bias sampling toward B^*.

Next, the `backup` function performs the standard vector backup operation (4.17) at selected nodes from $\mathcal{T}_R$. Even though new sampled points and α-vectors are generated with every `sample` and `backup` operations, not all of them are useful for constructing an optimal policy and are pruned to improve computational efficiency with the function `prune`.

Therefore, Sarsop is an any-time algorithm that returns the best policy found within a pre-specified amount of time. It gradually reduces the gap (average distance) between the upper and lower bounds on the value function at b_0, until it reaches either a pre-specified gap size or the time limit.

4.3 Literature about POMDPs

In this Section, previous works about POMDP techniques are reviewed. Different approaches to solve POMDPs and a wide variety of applications are cited. A basic presentation of the POMDPs for active perception is included in (Spaan, 2008). However, for a complete theoretical description the reader is directed to (Thrun et al., 2005; Kaelbling et al., 1998). In particular, the exact algorithm to solve the discrete POMDPs is shown in (Thrun et al., 2005), but it proves not to be efficient and several approximate methods are proposed instead.

4.3.1 Exact POMDPs

A POMDP for a specified finite horizon h can be solved optimally by using the value iteration algorithm described in (Sondik, 1971). This algorithm uses dynamic programming to compute increasingly more accurate values for each belief state. Smallwood and Sondik (1973) also show that the optimal value function for a finite-horizon POMDP can be represented by hyperplanes, and is therefore convex and piecewise linear (PWLC). This PWLC feature of the value function is exploited by most of the exact algorithms in the POMDP literature (Monahan, 1982; Littman, 1996; Cassandra et al., 1997; Zhang and Zhang, 2001).

The main challenge for these approaches is to cope with the number of α-vectors needed to represent the value function, which grows exponentially at each iteration. Besides, since the technique typically requires solving multiple linear programs to find candidate belief points where the value function is sub-optimal, the computation cost can be high. Furthermore, to guarantee that an exact solution is found, relevant beliefs must be generated systematically, meaning that all reachable beliefs must be considered. Therefore, these kinds of solvers can only be scaled for a reduced set of states/actions/observations.

Most of the work on exact approaches focuses on finding efficient manners to prune the set of α-vectors, as well as to effectively reduce computation. For instance, in (Zhang and Zhang, 2001), point-based updates are interleaved with standard dynamic programming updates to further accelerate value iteration. This approach is guaranteed to converge to the optimal solution.

4.3.2 Offline approximate POMDPs

Exact POMDP solvers are, in general, expensive algorithms which require offline computation. A POMDP is usually divided into two steps: first the optimal policy is calculated offline beforehand; and then, the best action is chosen and performed online at each iteration. Solving POMDP models offline can be implemented by computing a value function over the belief space, which defines a policy. Executing such a policy online is computationally cheap, but computing the value function can be very expensive.

Even though the first phase is performed offline and once, its computational cost may be exponentially high with the number of possible states. Due to such a high complexity, exact solvers are not feasible for large POMDPs and many researchers have worked on developing approximate offline approaches that can be applied to larger problems.

A first approach is to approximate the value function of the POMDP by the value function of its underlying MDP. This approach provides an upper bound on the

original value function. See for instance the well-known Q_{MDP} method in (Littman et al., 1995; Hauskrecht, 2000).

A second method is to approximate the value function in a grid-based manner, which means to build the value function from a finite set of belief points distributed in a grid and their corresponding values. These methods (Lovejoy, 1991; Bonet, 2002) create a fixed-resolution regular grid over the entire belief space, so that the value interpolations can be calculated quickly by considering only neighboring points of the grid. Zhou and Hansen (2001) also propose a grid with variable resolution by subsampling the regular grid proposed in (Lovejoy, 1991). Thus, some parts of the grid can be more densely sampled than others.

Although the grid-based approaches ignore the convexity of the value function, point-based methods (Pineau et al., 2003; Spaan and Vlassis, 2005; Smith and Simmons, 2004, 2005; Hauskrecht, 2000) take advantage of it and maintain a finite set of α-vectors to describe this value function. These are the most popular POMDP solvers and they approximate the value function by updating it only for some selected belief points. The set of belief points used can be obtained by simulating the POMDP model in different ways (Pineau et al., 2006). Then, the value function is just updated over those sampled beliefs, coping with the complexity of the exact solvers.

Pineau et al. (2003) propose a point-based value iteration (PBVI) algorithm that performs point-based backups only for beliefs that are reachable. The algorithm alternates between point-based backups and forward search to generate new reachable belief points. This technique tends to perform very well in practice, since for many POMDPs, the optimal value function of the reachable belief region can often be approximated well by a small set of α-vectors. Spaan and Vlassis (2005) propose a randomized version of PBVI called Perseus (see Section 4.2.2) that performs partial point-based backups by stopping early the generation of supporting vectors once the values of all the points have improved. Smith and Simmons (2004, 2005) also combine heuristic search with point-based value iteration to obtain a new algorithm called HSVI that simultaneously computes lower bounds (by point-based backups) and upper bounds (by heuristic search) on the optimal value function. Another point-based algorithm that maintains lower and upper bounds is Sarsop (Kurniawati et al.,

2008), which stands for Successive Approximations of the Reachable Space under Optimal Policies. Here, the key idea is to work with a belief tree that is expanded at each iteration with the more optimal policy calculated up to the moment. Thus, only more likely belief points are considered in the belief region, which is updated continuously. The algorithm was described in Section 4.2.4. Poupart (2005) presents an extension of Perseus. It is called *Symbolic* Perseus and uses Algebraic Decision Diagrams (ADDs) (Hoey et al., 1999) in order to optimize the operations in the original Perseus for factored POMDPs (see Section 4.2.3).

The above algorithms are all based on value iteration since they calculate iteratively an optimal value function by applying the Bellman Equation. Nevertheless, there exist a different kind of approaches based on policy iteration (Sondik, 1971, 1978; Hansen, 1997, 1998; Meuleau et al., 1999a,b). These methods search for the optimal solution directly within the space of policies.

Hansen´s classic policy iteration (Hansen, 1997, 1998) uses finite state controllers to represent policies. It incrementally improves the controller by alternating between two steps, policy improvement and policy evaluation, until converging to an optimal policy. Besides, the execution of policies represented by finite state controllers can be done in real time since there is no need for belief state monitoring. Ji et al. (2007) describe a point-based policy iteration (PBPI) algorithm for infinite-horizon POMDPs which replaces the exact policy improvement step of Hansen´s policy iteration with point-based value iteration (PBVI). Meuleau et al. (1999a) and Aberdeen and Baxter (2002) also propose gradient descent algorithms to find the best stochastic finite state controllers of a given size. In (Ng and Jordan, 2000), they search for tailored policies within a restricted class of compactly representable policies. When interested only in a policy for a given initial belief state, tailoring the policy to the belief region reachable from that initial belief state can be an effective way of reducing the complexity of policy representations. Thus, if the reachable belief region is a small subset of the entire belief space, tailored policies can be much simpler to find and represent than full policies. Poupart (2005) presents the *Bounded Policy Iteration* algorithm, which also optimize the original policy iteration (Hansen, 1997, 1998).

4.3.3 Online POMDPs

Online approaches (Ross et al., 2008) attempt to avoid the complexity of computing a complete policy by planning online only for the current information state, i.e. the most updated information the system has available at each moment. Whereas an offline approach would compute an exponentially large plan considering all possible situations, an online search only considers the current situation and a small horizon for the plans. Moreover, online approaches are suitable for many applications in which offline methods fail, since they allow real-time variations in the environment and the models. However, online methods also need to meet real-time constraints for computation, which is their major drawback.

Recent developments in online POMDP algorithms suggest that combining approximate offline and online approaches may be the most efficient way to tackle large POMDPs. In this case, a rough policy can be computed offline by means of any value iteration algorithm or equivalent, and then be used as a heuristic function to guide an online search algorithm. Thus, real-time constraints can be held without losing so much precision. Furthermore, the resulting time of running the offline and online parts of the algorithm could be reduced dramatically due to the combination of both.

An online algorithm is divided into a planning phase, and an execution phase, which are applied alternately at each time step. In the planning phase, based on the current belief state, the algorithm computes a tree with all the reachable belief states for a given horizon by considering the different available actions and observations. Then, an estimation of the value function at each belief from the tree can be made by means of an approximate value function computed offline (sometimes the algorithm maintains two value functions, an upper bound and an lower bound).

Once the planning phase terminates, the execution phase executes the best action found for the current belief in the environment, and updates the current belief and tree according to the observation obtained.

In (Paquet et al., 2005, 2006) those nodes from the tree that are known to be sub-optimal are pruned in order to prevent the expansion of unnecessary lower nodes and improve the efficiency of the search. Chang et al. (2004) propose expanding the tree by sampling a subset of observations at each expansion and only considering the

beliefs reached by these sampled observations. The algorithm uses a set of initial policies that may be calculated offline.

Ross and Chaib-draa (2007) try to focus the search on the most relevant reachable beliefs. The most relevant reachable beliefs are selected by means of heuristics so that the search is optimized, i.e. as few nodes as possible are expanded.

Another online approach, called SOVI (Shani et al., 2005), extends HSVI (Smith and Simmons, 2004, 2005) to an online value iteration algorithm without using any tree. This approach maintains a priority queue of the belief states encountered during the execution and performs α-vector updates for the current belief state and the belief states with highest priority at each time step. The priority of a belief state is computed according to how much the value function changed at successor belief states since the last time it was updated.

4.3.4 Hierarchical POMDPs

In many applications, the large number of states required to represent the environment makes the POMDP intractable or really hard to solve. Despite this, certain environments may be represented with different levels of resolution, depending on the application. For instance, when considering robot navigation indoors, each state can model a region of fixed size and the robot can take actions such as "turn-left", "turn-right" or "go-forward". Unfortunately, thousands of states could be necessary to model a real environment with a reasonable level of accuracy. However, for this example, a multi-resolution approach would fit perfectly. In such a case, there would be a mapping from abstract states (e.g whole corridors) to macro-actions (e.g, "go down the corridor") at an abstract level, whereas local plans ("exit a corridor" or "reach a goal location" within the corridor) would be produced at lower resolutions.

This is the main idea lying behind the Hierarchical POMDPs (Theocharous et al., 2001; Theocharous and Mahadevan, 2002; Theocharous et al., 2004). These Hierarchical POMDPs derive from the Hierarchical Hidden Markov Models (HHMM) (Fine et al., 1998) by adding primitive and abstract actions and reward functions. With this HHMM approach, there is a hierarchy of homogeneous representations of the

environment in which each layer of the hierarchy maintains a probabilistic model of the environment defined at a certain resolution. The tree structure of HHMM models also provides a natural approach for rapid model learning by reusing previously learned sub-models.

Theocharous et al. (2001) and Theocharous and Mahadevan (2002) generalize the HHMM to formulate the Hierarchical POMDPs and apply them to robot navigation in indoor environments. Theocharous et al. (2004) extend the previous works to represent the Hierarchical POMDPs by means of dynamic Bayesian Networks (DBNs). Pineau et al. (2001) propose to decompose the space of actions into a hierarchy of actions instead of a state-based decomposition. Toussaint et al. (2008) combine the Hierarchical POMDPs with a maximum-likelihood estimation technique for policy optimization. Foka and Trahanias (2007) compare some of the previous approaches to a novel Hierarchical POMDP technique aimed at navigating with an autonomous robot in real time.

The main problem of these approaches is that the hierarchical models must be created manually and there is not a method to compute them automatically yet. Due to this, it is more difficult to determine which are the abstract states, and that makes these approaches useful only for structured environments. Besides, further research is still needed to apply hierarchical models to other applications. In the literature, hierarchical POMDPs have been used to solve problems with thousands of states for single-robot navigation. However, many of the computation is done online, so adapting these approaches to multi-robot settings, where the number of states grows exponentially, is not straightforward.

4.3.5 MOMDPs

Some authors try to cope with the high dimensionality of the belief space by exploiting the idea that many robotic systems often have mixed observability, i.e. although the robot's state is not fully observable, some components may be. Thus, a factored model can be built to separate the fully and partially observable components of the state, leading to a lower-dimensional representation of the belief space.

This method is named Mixed Observability Markov Decision Processes (MOMDPs) and is proposed in (Ong et al., 2009) to perform some robotic tasks with large belief spaces. In order to overcome the computation problems derived from such large belief spaces, some components of the state are assumed to be fully observable (this is reasonable when there are available sensors which are accurate enough). Then, this new representation is used in conjunction with a point-based algorithm to compute approximate POMDP solutions. A related idea is applied to medical therapy planning (Hauskrecht and Fraser, 1998) for the diagnosis of ischemic heart diseases. Hsiao et al. (2009) also use an MOMDP for grasping objects robustly with a robot. They assume that the robot's position is approximately observable based on proprioception, and they do not include it in the uncertain part of the state. Therefore, a POMDP with fully and partially observable components is solved.

MOMDPs are used in some of the approaches proposed throughout this Thesis and will be further analyzed in Chapter 5.

4.3.6 Continuous POMDPs

Given the numerous optimization techniques for discrete models, a common approach for continuous models consists of discretizing or approximating the continuous components with a grid (Roy et al., 2005). This usually entails to trade off complexity for accuracy by varying the coarseness of the discretization. More precisely, as a discretization is refined, computational complexity increases.

In (Hoey and Poupart, 2005) a further step is taken by extending point-based value iteration to continuous observation spaces, using the fact that observations are useful only to the extent that they lead to different courses of action. The observation space can therefore be partitioned by calculating the thresholds at which different observations require different actions.

In the case of complete continuous POMDPs, a highly detailed work is done in (Porta et al., 2006). The algorithm Perseus (Spaan and Vlassis, 2005) is extended for solving continuous state POMDPs with discrete sets of actions and observations. The value function to optimize is proved to be PWLC over the belief space when

the beliefs are represented by means of particles or Gaussian Mixtures. Furthermore, reward, observation and transition models are also represented by linear combinations of Gaussians. At the end, the problem of continuous observations and actions is also tackled by introducing sampling methods that approximate the whole sets of actions or observations, respectively.

The work by Porta et al. (2006) is a generalization of (Duff, 2002), where a special case of continuous POMDPs in the context of model-based Bayesian reinforcement learning is considered, in which beliefs are maintained over the space of unknown parameters of the transition model of a discrete-state MDP. As those parameters are probabilities, the corresponding POMDP is defined over a continuous domain. Duff (2002) demonstrates that, for this special case, the optimal value function of the POMDP is parametrized by a set of functions, and for finite horizon it is piecewise-linear and convex (PWLC) over the belief space of multinomial distributions.

On the contrary, in (Brooks et al., 2006) probability distributions over the state space (beliefs) are represented in parametric form using low-dimensional vectors of sufficient statistics. The belief space, over which the value function has to be estimated, has dimensionality equal to the number of sufficient statistics. This work provides a dimensionality reduction and an efficient closed-form solution for belief updates. However, the value function is no longer PWLC, so a fitted value iteration algorithm is required.

4.3.7 POMDP applications

Over the years, there have been numerous examples demonstrating how POMDPs can be used for robot localization and navigation (Theocharous and Mahadevan, 2002; Theocharous et al., 2004; Roy et al., 2005). In particular, Foka and Trahanias (2007) propose a hierarchical model to perform robot navigation, localization and obstacle avoidance in an integrated manner. Results solving a POMDP in real time as well a as comparison to other hierarchical approaches are shown. The limitations of the hierarchical models have been stated above: they only suit highly structured environments. In (Foka and Trahanias, 2007), even though the complexity of the

models is alleviated by means of hierarchical models and the number of states is increased with respect to previous approaches, they assume that the observation and transition models are not dependent on the environment, which makes the approach unfeasible for active perception. Besides, the extension of the method to multi-robot settings is stated as future work.

There have also been applications for active sensing, which is highly related to the problem tackled in this Thesis. For instance, Darrell and Pentland (1996) propose a visual gesture recognition system, in which a POMDP controller steers the focus of the camera toward regions in the image which are most likely to improve recognition performance. Guo (2003) describes a POMDP framework for active sensing in which the actions are using a particular sensor (with an associated cost) or, when enough information has been gathered, outputting a particular classification label. Ji and Carin (2007) consider a similar setting, but they couple it with the training of HMM classifiers. In a multi-agent setting, Nair et al. (2005) and Varakantham et al. (2007) consider a distributed sensor network in which each sensor has to choose its gaze direction, in order to track targets. They can cope with teams of more than two agents, but the number of states remains quite reduced.

In a more recent work (Hoey et al., 2010), a very large POMDP is solved in order to develop a real-world system designed to assist elderly persons with cognitive deficiencies to carry out simple daily tasks, such as hand-washing. This is a remarkable approach for models that have an implicit structure, since POMDPs can be factorized. However, for a general application, it is not always straightforward to find these kinds of factorized models.

4.3.8 Multi-agent models

All the previous work in this Chapter was focused on single-agent systems, i.e. systems in which there is a sole agent that can take actions and gather information from the environment. That agent is the one in charge of estimating the state. However, this Thesis focuses on multi-robot systems, in which there are different agents that can perform cooperative actions, and take observations from the environment through

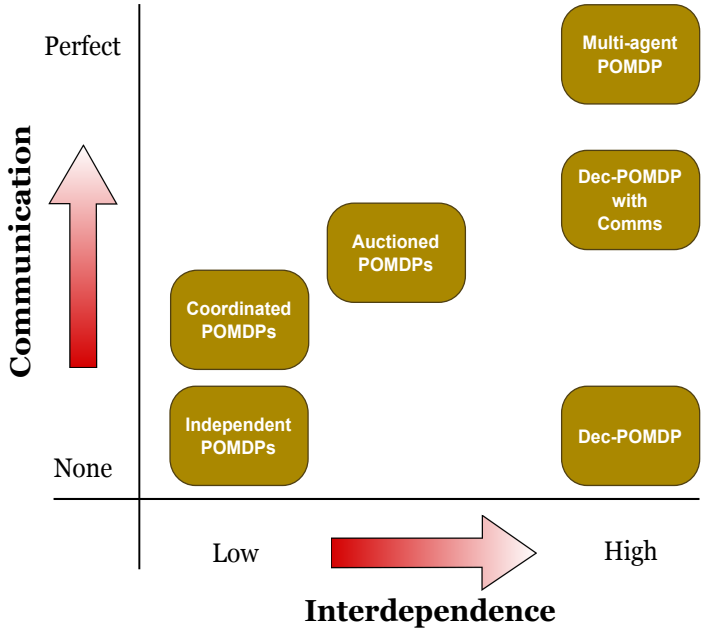


Figure 4.2: Classification of multi-agent POMDP approaches according to interdependence and level of communication between the agents. "Coordinated POMDPs" refers to the approach proposed in Chapter 5, and "Auctioned POMDPs" to the approach proposed in Chapter 6.

their sensors. The state is estimated by the whole set of robots, so all their actions and observations may affect a model of the system. The main problem of this multi-agent systems is that, unlike the single-agent systems, the dimensions of the action and observation spaces increase exponentially with the number of robots. Therefore, the complexity of solving POMDPs that consider all the mentioned variables is not always straightforward.

In the literature, there exists a wide variety of decision-theoretic models that extend POMDPs to deal with multi-agent systems (Seuken and Zilberstein, 2008), such as Multi-agent POMDPs (Pynadath and Tambe, 2002) and Decentralized POMDPs (Bernstein et al., 2002), which can be compared in terms of agent interdependence and communication assumptions. The level of interdependence between agents is determined by: (i) the amount of information that an agent needs to know about other agents; and (ii) how coupled the final policies are. Thus, a system is highly interdependent if a change in one of the agents' model requires re-computing the

policies for all the other agents. Many models from the literature are highly interdependent, for instance Multi-agent POMDPs (MPOMDP), Decentralized-POMDPs (Dec-POMDPs), Network Distributed POMDPs (ND-POMDPs) (Nair et al., 2005), and Iterative-POMDPs (I-POMDPs) (Gmytrasiewicz and Doshi, 2005).

Figure 4.2 presents a possible classification of existing multi-agent models with respect to their interdependence and the grade of communication that is assumed for the agents. The simplest approach is to map the global task as well as possible into a set of individual tasks, and model these as independent POMDPs (see Figure 4.2, bottom left). Thus, each agent can solve its own POMDP and execute its own policy without any communication. In this case, the interdependence between agents is very low, but since each agent ignores the others, there is no cooperation or even coordination either. Many interesting multi-agent planning problems cannot be tackled adequately with such a loosely coupled approach.

On the other hand, MPOMDPs and Dec-POMDPs solve a single decision-theoretic model for the whole team reasoning about all the actions and observations of each agent (see Figure 4.2, right column). The MPOMDP model assumes perfect communication and each agent has access to joint actions and observations at every moment, whereas the Dec-POMDP model assumes no communication at all. Such models achieve a tight cooperation, but they present a high interdependence, since any small change in one of the agents entails a recalculation of the policies for the whole team. Furthermore, if due to imperfect communication agents do not have access to other agents' observations, the behavior of the MPOMDP model is not defined. The Dec-POMDP model, on the other hand, does not exploit communication at all, which in many scenarios could be beneficial to improve team performance.

Due to the fact that no communication is assumed, Dec-POMDPs become complex models hard to solve (NEXT-complete problem). In this sense, most of the current algorithms for planning in Dec-POMDPs are based on some version of policy search (Szer et al., 2005; Varakantham et al., 2007; Bernstein et al., 2005). Nonetheless, some applications have been developed with Dec-POMDPs for simple domains. For instance, in (Emery-Montemerlo et al., 2005) the viability of approximate Dec-POMDP techniques for controlling a small group of robots is demonstrated, whereas

in (Becker et al., 2004) an application about multi-robot space exploration is addressed.

Other works are concerned about whether similar value functions than the ones in the single-agent case (Porta et al., 2006) can be defined for Dec-POMDP models, and how policies can be extracted from such value functions. In (Oliehoek et al., 2008), two different forms for the optimal value function in Dec-POMDPs are defined: one that gives a normative description as the value function of an optimal pure joint policy and another one that is sequentially rational and thus gives a recipe for computation. This computation, however, is unfeasible for all but the smallest problems, so various approximate value functions that enable an efficient computation are analyzed.

Game theory can also be applied to the multi-agent decentralized problem. In (Emery-Montemerlo et al., 2004) a Partially Observable Stochastic Game (POSG) with common pay-offs is proposed instead of Dec-POMDPs. Due to its intractability, the Partially Observable Stochastic Game is approximated by a set of Bayesian games.

In between MPOMDPs and Dec-POMDPs there are several models in which some communication is assumed (Nair et al., 2004; Roth et al., 2005; Spaan et al., 2008). These models try to exploit the fact that agents actually share information, but just partially and at certain instants. Furthermore, most of them assume that communication arrives instantly.

4.3.9 Literature classification

Throughout this Section, several criteria have been used in order to classify the wide range of different POMDP approaches. Even though multiple classifications are possible, the most common areas in the existing literature have been addressed. Thus, exact POMDP solvers as well as approximate approaches have been cited. Then, within the approximate methods, offline and online techniques were differentiated. Finally, works on continuous POMDPs, hierarchical POMDPs, MOMDPs and multi-agent models have also been presented. All these previous works are classified in Table 4.1 by grouping them according to the explained categories.

4.4 Conclusions

This Chapter presented POMDP models for systems based on a single agent or multiple agents. It also discussed how these models can be used for active perception. An extensive review of the literature was provided, including different multi-agent models and active perception applications.

The main goal of this Thesis is to perform active perception with a team of multiple robots; the multi-robot model described in Section 4.1.4 suits such purpose. However, as it was highlighted throughout the Chapter, this model cannot scale in line with the number of robots, since its complexity grows exponentially as more agents are included. In addition, the dimension of the state space itself introduces a second source of complexity and, as the environment size grows, the model may become intractable. Hence, these two features make the model feasible only for domains with a few states and agents.

Similar problems were noted in Section 4.3.8 for other multi-agent systems. Therefore, in Section 4.1.4 the Thesis proposes approximations of the initial model in order to apply the methods to larger domains in robotic applications. In general, the objective is to exploit the power of decision-theoretic multi-agent methods, but keeping in mind the possibilities and constraints posed by multi-robot systems. Of particular relevance in this context is the fact that communication between robots is often possible, but the quality of the communication channel can vary. This precludes centralized solutions as well as methods requiring communication guarantees.

A first approximation is to consider mixed observability in the system. That notion was introduced in Section 4.3.5 and leads to a severe reduction in the complexity of the models for certain applications. Chapter 5 will detail how to alleviate high-dimensional domains under the assumption of mixed observability and how this approximation suits the robotic applications considered in this Thesis.

Another approximation is to separate the initial MPOMDP model into several decoupled models which are less complex, but can be used together to emulate the optimal behavior derived from the original model. In this direction, two different approaches will be presented in Chapters 5 and 6. Recalling the classification of

multi-agent models in Section 4.3.8, the method proposed in Chapter 5 appears in Figure 4.2 as "coordinated POMDPs", since it builds on the "independent POMDPs" but adding some additional exchange of information. Besides, the term "auctioned POMDPs" refers to the approach described in Chapter 6, which adds a properly cooperative layer to the system by auctioning POMDP policies.

Approach		**References**
Exact POMDPs		(Sondik, 1971; Smallwood and Sondik, 1973; Monahan, 1982; Littman, 1996; Cassandra et al., 1997; Zhang and Zhang, 2001; Guo, 2003)
Offline approximations	MDP	(Littman et al., 1995; Hauskrecht, 2000)
	Grid-based	(Lovejoy, 1991; Bonet, 2002; Zhou and Hansen, 2001)
	Point-based value iteration	(Pineau et al., 2003; Spaan and Vlassis, 2005; Kurniawati et al., 2008; Smith and Simmons, 2004, 2005; Hauskrecht, 2000; Pineau et al., 2006; Poupart, 2005; Hoey et al., 2010; Ji and Carin, 2007)
	Policy iteration	(Sondik, 1971, 1978; Hansen, 1997, 1998; Meuleau et al., 1999a,b; Ji et al., 2007; Aberdeen and Baxter, 2002; Poupart, 2005; Ng and Jordan, 2000)
Online approximations		(Ross et al., 2008; Paquet et al., 2005, 2006; Chang et al., 2004; Ross and Chaib-draa, 2007; Shani et al., 2005; Darrell and Pentland, 1996)
Hierarchical POMDPs		(Theocharous et al., 2001; Theocharous and Mahadevan, 2002; Theocharous et al., 2004; Pineau et al., 2001; Toussaint et al., 2008; Foka and Trahanias, 2007)
MOMDPs		(Ong et al., 2009; Hauskrecht and Fraser, 1998; Hsiao et al., 2009)
Continuous POMDPs		(Roy et al., 2005; Hoey and Poupart, 2005; Porta et al., 2006; Duff, 2002; Brooks et al., 2006)
Multi-agent models	MPOMDPs	(Seuken and Zilberstein, 2008; Pynadath and Tambe, 2002)
	Dec-POMDPs	(Bernstein et al., 2002; Emery-Montemerlo et al., 2005; Becker et al., 2004; Szer et al., 2005; Varakantham et al., 2007; Bernstein et al., 2005; Oliehoek et al., 2008)
	Dec-POMDPs with communications	(Nair et al., 2005; Gmytrasiewicz and Doshi, 2005; Nair et al., 2004; Roth et al., 2005; Spaan et al., 2008)
	POSGs	(Emery-Montemerlo et al., 2004)

Table 4.1: Classification for previous works on POMDPs.

Chapter 5

Multi-robot coordinated decision making under mixed observability

This Chapter proposes a coordinated approach for multi-robot teams in order to deal with decision-making under uncertainty. In particular, the decentralized data fusion scheme developed in Chapter 2 is combined with a POMDP-based framework in order to add decision capabilities to the system. As it was seen in Chapter 4, the complexity of POMDPs may increase remarkably when the system considers multiple agents or large state spaces, turning them unfeasible for some real domains. This Chapter attempts to produce a scalable approach by tackling both sources of complexity: the large domains in real applications and the inclusion of multiple robots.

Firstly, the concept of *mixed observability* and its implications are introduced for POMDPs in order to alleviate large domain spaces,. *Mixed observability* may lead to simpler representations of the problem by assuming that some of the components in the state are fully observable, and solve the POMDP only on the partially observable part. Moreover, some of the main policy solvers in Chapter 4 are revised in order to incorporate this new concept. It will also be shown that this assumption is reasonable within a wide range of robotic applications.

Later, the non-scalability of multi-robot POMDPs is addressed by using a decentralized approach in which robots have no knowledge about others' actions. In this situation, an implicit coordination will be derived from sharing the beliefs among the

robots through decentralized data fusion. This coordinated approach is applied to a tracking application with multiple robots.

5.1 Introduction

Techniques for planning under uncertainty are being applied more and more to robotics (Vlassis et al., 2006; Prentice and Roy, 2009). As it was shown in previous Chapters, POMDPs offer a robust mathematical framework but do not have scalability. This is because the optimal plan has to be searched on the belief space, which can be very large. This drawback is even more evident in multi-robot teams, as the space of actions and observations increases exponentially with the number of robots.

In Chapter 4, several approaches aimed at alleviating such sources of complexity were introduced. In particular, approximate point-based methods to obtain POMDP policies were presented (see Section 4.2). In Section 4.3.5, it was also highlighted how other authors (Ong et al., 2009) try to cope with the high dimensionality of the belief space by exploiting the idea that many robotic systems often have mixed observability, i.e. although the state is not fully observable, some components might be. Thus, the state can be factorized leading to a lower-dimensional representation of the belief space.

Although these techniques can cope with larger belief spaces, it was shown that scalability in line with the number of robots in a team (computing a global plan) is still lacking. Most existing multi-robot models (see Section 4.3.8) either do not exploit communication among the members of the team (Dec-POMDPs) or increase exponentially their complexity with the number of robots (MPOMDPs).

Therefore, even though the MPOMDP model for multi-robot systems was proposed to solve the problem of active perception in Chapter 4, it may become intractable for real applications. This Chapter combines two methods to reduce the complexity of that original model and obtain a practical solution. First, MOMDPs are used to consider mixed observable states and reduce the dimension of the belief

space. Although the approach is tested for cooperative tracking, it will be shown how these models can fit for a wider range of robotic applications.

A further idea exploited in this Chapter is that communication and data fusion between the robots can induce a common Markovian belief to coordinate the robots plans, alleviating the second source of complexity mentioned above and scaling the model with the number of robots. Hence, instead of solving the complete MPOMDP for the joint belief (including information about the actions and observations from all the robots), this model is separated into a local POMDP for each robot that is solved with its local belief (which takes into account local observations only). However, the policies obtained from these local POMDPs are executed over a joint belief that considers information from all the team members. This joint belief is calculated through a Decentralized Data Fusion (DDF) algorithm, which induces a common belief among the robots and coordinates the plan execution. Even though this approach does not guarantee a global optimal policy (since actions from the whole team are not considered when planning for each robot), it produces an implicit coordination between the robots and, more importantly, it turns an intractable planning problem into a solution that can be scaled depending on the number of robots.

The idea is similar to the one proposed in (Grocholsky et al., 2003; Mathews et al., 2008), where a distinction between cooperative and coordinated information-theoretic approaches is made, proposing the latter to control fleets of robots. In a coordinated approach, the members of the team have no knowledge about the others' models or control actions, but they exchange some information that may influence implicitly other members' subsequent decisions. Therefore, a coordinated behavior can be obtained by sharing fused perception information or the impact of others' control actions over a certain objective function and acting locally. However, this approach and others such as the ones in (Wong et al., 2005; Bourgault et al., 2004), use greedy control algorithms in order to determine the most suitable coming action but do not reason about long-term goals, unlike the approach proposed in this Chapter.

5.2 Mixed observability Markov decision processes

In a POMDP, even for a finite set of $|S|$ states, the policy π is defined over an $(|S|-1)$-dimensional continuous belief space. Hence, high-dimensional belief spaces are a remarkable source of complexity when solving POMDPs. The idea of considering mixed observability is to model these kinds of systems with a factored POMDP where the fully and not fully observable parts are separated. This model is called a Mixed Observability MDP (MOMDP) and can be combined with a point-based solver to solve traditional POMDPs (Ong et al., 2009).

Intuition says that extracting the fully observable part, the resulting POMDP that has to be solved has lower dimensionality. For instance, let imagine a tracking application where there is a target who has to be tracked by a pursuer robot. The robot is equipped with on-board sensors aimed at estimating the position of the target in order to follow it. If the work area is approximated with a grid, this problem can be modeled by a discrete POMDP, where the state consists of the locations of the target and the robot. In case the grid were 10×10 cells and the robot had 4 different orientations, there would be 400 possible locations for the pursuer and 100 for the target, which means a 40,000-dimensional belief space. However, in this example the actual uncertainty could lie behind the target position, whereas the robot pose could be determined with enough accuracy by means of the on-board sensors. Hence, assuming the robot pose known, the belief space becomes a union of 400 disjoint 100-dimensional subspaces.

Therefore, the key in MOMDPs is to gain computational efficiency by solving a number of lower-dimensional POMDPs instead of the original one. Thus, all the operations for any solver have to work on lower-dimensional belief spaces, which can lead to a remarkable improvement in the performance.

5.2.1 MOMDP model

The mathematical model for a MOMDP is quite similar to the original POMDP and it is introduced by Ong et al. (2009). The MOMDP is represented as a factored POMDP where the state vector is composed of two different parts. Component x is

the fully observable part of the original state s, and y is another vector representing the partially observable part. Thus, the state is specified by $s = (x, y)$, and the state space is $S = X \times Y$, where X is the set with all the possible values for x and Y all the possible values for y. For instance, in the previous tracking example, y would be the unknown position of the target whereas x would be known position of the pursuer robot.

Formally, a MOMDP is defined by the tuple $\langle X, Y, A, Z, T_x, T_y, O, R, h, \gamma \rangle$. The components are the same as in the POMDP case, but the transition function T is now decomposed into T_x and T_y. $T_x(x', a, x, y) = p(x_t = x' | a_{t-1} = a, x_{t-1} = x, y_{t-1} = y)$ gives the probability that the fully observable state component has value x' if the robot takes action a at state (x, y). $T_y(y', x', a, x, y) = p(y_t = y' | x_t = x', a_{t-1} = a, x_{t-1} = x, y_{t-1} = y)$ gives the probability that the partially observable state component has value y' if the robot takes action a at state (x, y) and the fully observable state component has value x'.

In a MOMDP, since it can be observed, there is no need to maintain a belief over x. Therefore, it can be excluded in order to just maintain a belief $b_y(y)$, which is a probability distribution over y. Any belief $b \in B$ from the complete system state $s = (x, y)$ is then represented as (x, b_y), where $b_y \in B_y$. Furthermore, for each value x of the fully observable state component, a belief space $B_y(x) = \{(x, b_y) | b_y \in B_y\}$ is associated. Hence, each $B_y(x)$ is a subspace in B, and B is the union of these subspaces $B = \bigcup_{x \in X} B_y(x)$.

Note that while B has $|X||Y|$ dimensions, each $B_y(x)$ has only $|Y|$ dimensions. Therefore, the objective of representing a high-dimensional space as a union of lower-dimensional subspaces is achieved. Moreover, when $|Y|$ is small, a remarkable computational improvement can be reached, since operations to solve the MOMDP are performed over the subspaces $B_y(x)$.

Now, since every belief is represented by (x, b_y), Ong et al. (2009) prove that the value function can be described as:

$$V(x, b_y) = \max_{\alpha \in \Gamma_y(x)} \alpha \cdot b_y \tag{5.1}$$

where for each x, $\Gamma_y(x)$ is a set of α-vectors defined over $B_y(x)$. Therefore, the value function is now a collection of sets of $|Y|$-dimensional vectors, and the value of the observable component x determines which $\Gamma_y(x)$ is selected. Then, the vector from $\Gamma_y(x)$ that maximizes the value function is calculated. Since the vectors have $|Y|$ dimensions instead of $|X||Y|$, the execution of the policy is also faster for a MOMDP.

5.2.2 Sarsop under mixed observability

Once MOMDPs have been introduced, the question is how to solve them. Fortunately, with some modifications, the same point-based algorithms for POMDPs are valid now. Since the value function consists of a set of α-vectors for every $x \in X$, the idea is to run an independent value iteration for each of them separately. In (Ong et al., 2009), Sarsop (see Section 4.2.4) is adapted in order to cope with mixed observability.

The two main steps to adapt point-based POMDP solvers to deal with MOMDPs are the belief update and the backup operation. In a POMDP, the belief update was determined by Equation (4.3). However, $b_y(y) = b(s)$ when $s = (x, y)$ in a MOMDP, so (4.3) can be rewritten as:

$$b^z_{a,y}(y') = \eta p(z|a, x', y') \sum_{y \in Y} p(x', y'|a, x, y) b_y(y) \tag{5.2}$$

$$= \eta p(z|a, x', y') \sum_{y \in Y} p(y'|x', a, x, y) p(x'|a, x, y) b_y(y) \tag{5.3}$$

$$= \eta O(z, a, x', y') \sum_{y \in Y} T_y(y', x', a, x, y) T_x(x', a, x, y) b_y(y) \tag{5.4}$$

When the belief is updated, the values of the observable states before (x) and after (x') taking an action are known. Hence, (5.4) is particularized for specific values of those variables.

The other major modification in MOMDP solvers is the backup operation. Based on all the considerations made for MOMDPs, (4.14) can be rewritten as:

$$\begin{aligned} V_n(x, b_y) = \max_a \ [\ R_a^x \cdot b_y + \gamma \sum_{z \in Z} p(z|a, x, b_y) \\ \times \max_{\alpha \in \Gamma_{y,n-1}(x')} \sum_{x' \in X} \sum_{y' \in Y} b_{a,y}^z(y') \alpha(y') \ ] \end{aligned} \tag{5.5}$$

Note that $R_a^x(y) = R(x, y, a)$. In this case, unlike (5.4), all the possible x' must be taken into account. Even though x' and z are concrete values, they cannot be known beforehand like x, so all their values must be considered and weighted by their probabilities. Thus, developing (5.5) as it was done for (4.14), the backup operator can be transformed into:

$$\texttt{backup}(b_y) = \arg\max_{a \in A} \ b_y \cdot \alpha_a, \text{ where} \tag{5.6}$$

$$\begin{aligned} \alpha_a(y) = & R_a^x + \gamma \sum_{z \in Z} \sum_{x' \in X} \sum_{y' \in Y} (T_x(x', a, x, y) \\ & T_y(y', x', a, x, y) O(z, a, x', y') \alpha_{a,x',z}(y')) \end{aligned} \tag{5.7}$$

Finally, note that:

$$\alpha_{a,x',z}(y') = \arg\max_{\alpha \in \Gamma_{y,n-1}(x')} \ \alpha(y') \cdot b_{a,y}^z(y') \tag{5.8}$$

Therefore, the original Sarsop algorithm (see Section 4.2.4) can be adapted to include mixed observability. The new version is summarized in Algorithm 6. Now, a lower bound $\underline{V}$ and an upper bound $\overline{V}$ on the value function (5.5) are initialized and maintained for each $x \in X$. Besides, the belief tree $\mathcal{T}_R$ is made up of mixed nodes (x, b_y). The `sample`, `backup` and `prune` functions still play the same role as they played in the original Sarsop. However, the belief update and the backup operator equations are altered as it was explained above.

Algorithm 6 MO-Sarsop($X, Y, A, Z, T_x, T_y, O, R, h, \gamma$)

for each $x \in X$ **do**
  Initialize $\Gamma_{y,0}(x)$, representing the lower bound $\underline{V}$. Initialize the upper bound $\overline{V}$.
end for
Insert the initial belief point (x_0, b_{y0}) as the root of the tree $\mathcal{T}_R$
$n = 0$
repeat
  `sample`($\mathcal{T}_R$,Γ_n)
  Choose a subset of nodes from $\mathcal{T}_R$. For each chosen node (x, b_y), `backup`($\mathcal{T}_R$,Γ_n,(x, b_y))
  `prune`($\mathcal{T}_R$,Γ_n)
  $n = n + 1$
until Gap between $\underline{V}$ and $\overline{V}$ is small enough or time limit is reached.
return Γ_n

5.2.3 Symbolic Perseus under mixed observability

In this Section Symbolic Perseus (see Section 4.2.3) is extended to cope with mixed observability. The resulting algorithm is named MO-Symbolic Perseus and it provides the means for a useful comparison with Sarsop for MOMDP users. Moreover, recalling that Symbolic Perseus exploits the factored representations of POMDPs and the ADD structures in order to optimize the original Perseus, MO-Symbolic Perseus can be of great interest in certain domains where the variables are not very coupled.

As stated in the previous Section, there are two basic steps to modify: the backup operator and the belief update. These variations are the same in Sarsop and MO-Symbolic Perseus.

Algorithm 7 describes the whole procedure. Some modifications are needed when initializing (lines 2 and 3). First, a different set of beliefs $B_y(x)$ over the component y must be sampled for each value of x. In case y is probabilistically independent of x, the same set of beliefs over y could be used for every x. All the value functions $\Gamma_y(x)$ are also initialized separately with the same values as in the original Symbolic Perseus. The rest of the procedure is similar to the original Symbolic Perseus but being repeated for every $x \in X$. Moreover, all the operations are made with vectors of $|Y|$ dimensions now, which can simplify some steps.

Algorithm 7 MO-Symbolic Perseus$(X, Y, A, Z, T_x, T_y, O, R, h, \gamma)$

for each $x \in X$ **do**
  Sample set of reachable beliefs $B_y(x)$
  Initialize $\Gamma_{y,0}(x) = \{R_a^x | a \in A\}$
end for
$n = 0$
repeat
  $n = n + 1$
  for each $x \in X$ **do**
    $\Gamma_{y,n}(x) = \emptyset$
    $\tilde{B} = B_y(x)$
    while $\tilde{B} \neq \emptyset$ **and** $|\Gamma_{y,n}(x)| < MAXSIZE$ **do**
      Sample $b_y \in \tilde{B}$
      $\tilde{B} = \tilde{B} \setminus \{b_y\}$
      $\alpha_y^* = \texttt{backup}(b_y)$
      if $\Gamma_{y,n}(x) = \emptyset$ **or** $\max_{b_y \in B_y(x)}(b_y \cdot \alpha_y^* - V_n(x, b_y)) > 0$ **then**
        $\Gamma_{y,n}(x) = \Gamma_{y,n}(x) \cup \{\alpha_y^*\}$
      end if
      $\tilde{B} = \{b_y \in \tilde{B} : V_n(x, b_y) < V_{n-1}(x, b_y)\}$
    end while
  end for
until convergence
return $\Gamma_{y,n}(x)$ $\forall x$

5.3 Coordination through decentralized data fusion

Given a multi-robot planning problem, a possible option is to solve a MPOMDP for the whole team, considering all the potential joint actions and observations, but this approach ultimately cannot be scaled in line with the number of robots. Without losing generality, in some applications the reward function of the MPOMDP could be decomposed into an addition of N reward functions, one for each of the robots: $R(s, a^J) = R^1(s, a^J) + \cdots + R^N(s, a^J)$. All these reward functions would be defined over the joint actions and the joint belief, which is the one that includes information about the joint observations and actions. Moreover, the MPOMDP could be separated into N POMDPs, each of them with one of the previous reward functions. These

POMDPs could be solved and executed separately for each robot considering only their local beliefs, instead of the joint belief. However, if the robots solve independent POMDPs, and use only their local information (action and observations), the sub-optimal policies would be executed over different belief states for each robot. On the other hand, if communication is allowed among the robots and the joint belief state can be recovered locally (as it was done in Chapter 2), this would represent a coordination signal that summarizes all the information gathered by the fleet. That solution should be sub-optimal regarding the fully cooperative centralized solution (MPOMDP), but it can represent a trade-off between quality and complexity.

Therefore, for a team of N robots, a decentralized scheme is proposed, where each robot i solves its own POMDP without considering the others' actions or observations, i.e. with its local belief and its local reward. Then, once the policies have been calculated, they are executed over the joint belief. Thus, the coordination is achieved during the execution phase by sharing a common belief state $b_{cen}(s)$ that considers information from all the robots. In particular, if $a^J = \langle a^1, \cdots, a^N \rangle$ is the joint action and $z^J = \langle z^1, \cdots, z^N \rangle$ the joint measurement, a centralized node with access to all the information would update the belief according to:

$$b_{cen}(s') = \eta O(z^J, a^J, s') \sum_{s \in S} T(s', a^J, s) b_{cen}(s) \tag{5.9}$$

Chapter 2 of this Thesis tackles the problem of recovering this centralized belief in a decentralized manner. Assuming that the data gathered by the different robots at any time instant are *conditionally independent* given the state at that instant s', i.e. $p(z^J|a^J, s') = \prod_i p(z^i|a^i, s')$ (local measurements do not depend on the others' actions or observations), it is possible to combine locally the received beliefs from other robots with the local one of robot i, $b^i(s)$, to recover a common belief. For this purpose, the common information between each pair of robots i and j (i.e. information previously exchanged between the robots), which is represented by $b^{ij}(s)$, can be maintained by a separate filter called Channel Filter (Bourgault and Durrant-Whyte, 2004). The rules to fuse the beliefs from different robots and update the Channel Filters were developed in Chapter 2 for the Gaussian case (assuming tree-network topologies), but

they can also be derived for discrete filters. Algorithm 8 describes a decentralized version of the grid-based filter that is used here to reach a common belief among the team of robots. Basically, it is the same one as in the continuous case but with discrete-based filters.

In the line 1 of the Algorithm, the belief for robot i is updated with its local measurements. Unlike Chapter 2, the robots are controlled, so their actions must be considered when updating the belief as long as the state depends on the actions. The common information for each link must also be predicted in order to refer to the same time instant (line 3). Note that if the state predictions (lines 1 and 3) depend on others' actions too, a^J should be used at each agent instead of their local actions. Otherwise, the Channel Filter only gives an approximate solution. In lines 5 and 6, the information of the local belief is fused with the information received from the neighbors after removing the common information. Afterwards, the belief must be normalized so that the probabilities of the grid add up to 1. Finally, the fused information can be sent to other robots (line 7) and the Channel Filters updated (lines 10 and 11). In logarithmic form, this update only consists of adding to the previous common information the new information that was received or sent through the corresponding communication link.

In Chapter 2 is shown that it is possible to obtain locally the same belief as in a centralized node with access to all the information available, by including delayed states in the belief. If only the current state is considered, some information is lost with respect to an ideal centralized fusion unless the robots communicate every time they gather new information (Bourgault and Durrant-Whyte, 2004). This is the case here, where only information about the last time step is being considered for the grid-based filters. This information about past states can not be maintained for non-parametric filters in an efficient way, as it was in the Gaussian case. Therefore, it is not worthwhile anymore to keep full trajectories of the state, since that would increase the communication bandwidth significantly. In addition, the dynamics of the systems considered in this Thesis for the discrete case are lower than the communication rates, so delayed data are not so meaningful as they were in the continuous case, and robots can send information every time they gather it. Actually, the experimental results

Algorithm 8 $b^i(s') \leftarrow$ Decentralized grid-based filter for robot i ($b^i(s)$)

1: $b^i(s') = \eta_i O(z^i, a^i, s') \sum_{s \in S} T(s', a^i, s) b^i(s)$ {; Local update}
2: **for all** $j \neq i$ **do**
3: $\quad b^{ij}(s') = \eta_{ij} \sum_{s \in S} T(s', a^i, s) b^{ij}(s)$ {; Predict common information}
4: **end for**
5: $b^i(s') = b^i(s') \prod_{j \neq i} \frac{b^j(s')}{b^{ij}(s')}$ {; Update received information from neighbors}
6: $b^i(s') = \frac{b^i(s')}{\sum_{s' \in S} b^i(s')}$ {; Normalize}
7: Send $b^i(s')$ to neighbors
8: {; Update channel filters}
9: **for all** $j \neq i$ **do**
10: $\quad \log b^{ij}(s') = \log b^{ij}(s') + \underbrace{\log b^j(s') - \log b^{ij}(s')}_{j \to i} + \underbrace{\log b^i(s') - \log b^{ij}(s')}_{i \to j}$
11: $\quad b^{ij}(s') = \frac{b^{ij}(s')}{\sum_{s' \in S} b^{ij}(s')}$ {; Normalize}
12: **end for**

in this Chapter and the next one will show that these decentralized grid-based filters are quite reliable in terms of maintaining a common belief for the robots.

Note that instead of using grid-based filters, another option would be to use a decentralized Gaussian filter like the one in Chapter 2, and discretize its output afterward in order to feed the POMDP policy and obtain the optimal actions. In this case, a smoother representation of the state in continuous space would be obtained. Moreover, delayed states could still be used to deal with communication latencies. However, that filter would not consider belief distributions with multiple hypothesis, what would reduce dramatically the performance of the POMDPs. In order to exploit properly the advantages of POMDPs, rich beliefs are needed, what makes multi-hypothesis representations essential.

Finally, with the decentralized approach proposed, coordination arises implicitly due to the fused belief, and there is no need to solve a MPOMDP for the whole team, whose complexity grows exponentially with N. Of course, the policy is not optimal, as the robots do not reason about the other robot actions with respect to the global reward, but as it will be seen, through proper design of the rewards and the

communication between robots, it can be obtained a helpful coordinated behavior for many applications. Actually, this technique suits a wide range of problems in which individual behaviors have to be coordinated, such as surveillance (Hsieh et al., 2007), forest fire detection (Merino et al., 2006) or robotic soccer. Particularly, in this Chapter it is applied to multi-robot tracking.

5.4 Coordinated MOMDPs for target tracking

In order to illustrate the coordinated approach proposed in this Chapter, an application for tracking a target by means of multiple robots is considered. In this problem there is a moving target and a team of N robots which are the pursuers. Each of these robots carries a sensor which determines whether the target is visible or not within its field of view (FOV). Thus, the objective is to find the target in the environment and localize it as well as possible. This is clearly a problem of active perception, since the different robots have to act together in order to improve their estimation, which is the target position in this case.

The state is composed of the position of the target and the positions of the pursuer robots. The heading of each pursuer is also considered and included in the state. The state space is discretized into a cell grid, and a map of the scenario is assumed to be known. There are also four possible headings for each robot: *north*, *west*, *south* or *east*.

At each time step, each robot can choose between four possible actions: *stay*, *turn right*, *turn left* or *go forward*. *stay* means doing nothing; when *turning*, the robot changes its heading 90^o degrees; and when *going forward*, it moves to the cell ahead. Nonetheless, noisy transition functions for the states of the robots are considered. Besides, the target is assumed to move randomly. The transition function for its position indicates that, from one time step to the next, the target can move to any of its 8-connected cells with the same probability (only non-occupied cells are considered in order to calculate that probability).

In addition, each sensor provides a boolean measurement: *detected* or *non-detected*. These sensors proceed as it follows: if the target is out of its FOV, the sensor produces

a *non-detected* measurement; however, when the target is within its FOV, it can be *detected* with a probability p_D. The robots are heterogeneous in the sense that the sensor's FOV and p_D can vary from one robot to another.

Finally, a design of the reward functions so that the target is tracked by the team of robots is crucial. Since some cooperation is desirable within the heterogeneous team, a different behavior is assigned to each robot. Thus, the reward function also depends on the specific robot: $\{R^1(x, y, a), R^2(x, y, a), \cdots, R^N(x, y, a)\}$. Moreover, for all the members of the team, no cost is assigned to the action *stay*, whereas a cost of 1 is associated with the other actions.

This application is a fair example of how MOMDPs can help to reduce the belief space dimensionality. Here, even though robot and target locations are considered within the state, the one involving a greater uncertainty is the latter. Actually, the locations of the robots may be assumed to be observable (they could be obtained accurately enough by means of the on-board sensors and the available map, at least for the cell resolution), remaining as non-observable the target's position. Thus, the non-observable part of the state consists of the target location, $y = tar_l$, whereas the fully observable part consists of all the robots locations and headings, $x = (rob_l^1, rob_h^1, \cdots, rob_l^N, rob_h^N)$. For instance, for a 10x10-cell grid, there would be 400 possible states for each pursuer and 100 for the target, which means, for a single robot, to reduce a POMDP with a 40,000-dimensional belief space to a union of 400 disjoint 100-dimensional subspaces.

Besides, the proposed coordinated approach will be applied. That means that each robot i solves its own MOMDP without considering the other robots. That MOMDP has states $x^i = (rob_l^i, rob_h^i)$ and $y = tar_l$, and reward $R^i(x^i, y, a^i)$, being possible to specify a different strategy for each robot. Thus, once the policies have been calculated, coordination arises during the execution phase by sharing the fused belief state that considers information from all the robots. Of course, the optimal solution would be that obtained by a MPOMDP whose reward function is an addition of the rewards of each robot, but defined over the joint action and state. Thus, the reward of each robot would also depend on the actions of the others. Therefore, it is important to remark that this decomposition of the main reward function into

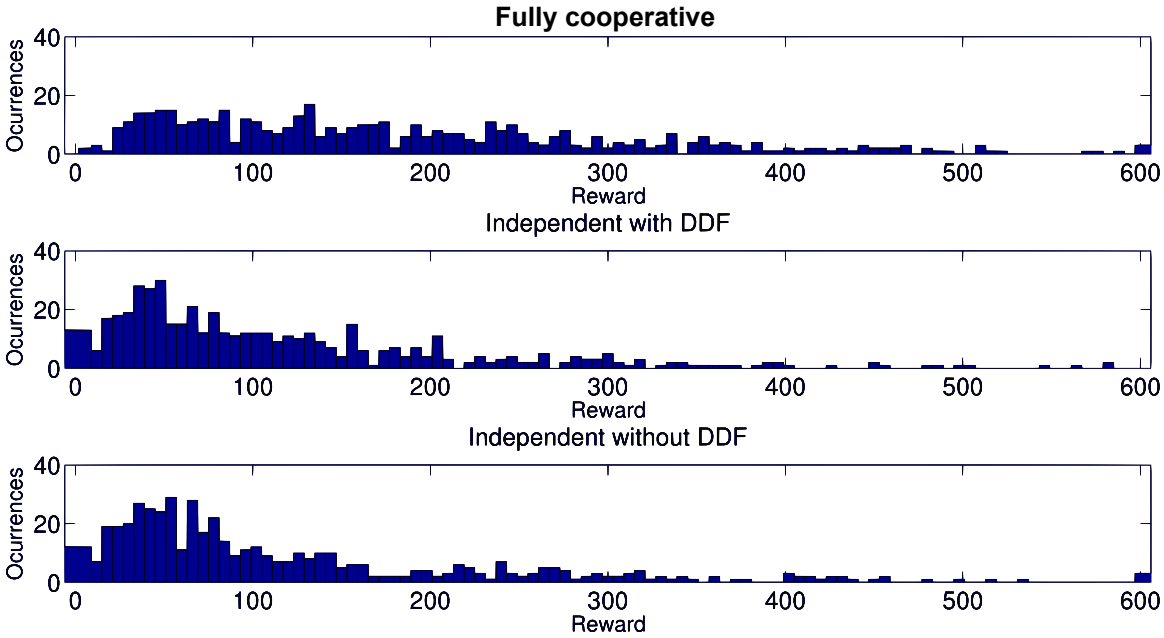


Figure 5.1: Histograms of the average rewards obtained during 500 simulations for the different approaches. At each simulation the expected discounted reward for the whole team is considered.

independent ones with only local information will lead to a sub-optimal solution. Nonetheless, the objective of the approach proposed here is to maximize the total reward for the whole team (i.e. the addition of the individual rewards for each robot) getting as close as possible to the optimal solution given by the MPOMDP.

5.5 Experiments

5.5.1 Simple scenario

A simple target tracking scenario was simulated with Matlab in order to highlight the differences between the proposed approach and a fully cooperative centralized solution. The second solution is optimal, since a MPOMDP is solved for the whole team considering actions and observations from all the robots. In both cases, the model is the one explained previously and particularized for a map with 11 available cells (a 3x4 grid with an obstacle in the middle). Then, two pursuer robots are

considered with equal FOVs, composed by a single cell in front of them and $p_D = 0.9$. Each robot gets a high reward (+100) when the target is in the cell of its FOV, otherwise the reward is zero. The discount factor is $\gamma = 0.9$.

In the coordinated approach proposed in this Chapter, a MOMDP for each single robot without considering the other were solved. The fully cooperative case was solved with a single MOMDP that considered actions and observations from both robots (basically a MPOMDP incorporating mixed observability). The reward for that second model was simply the sum of the independent rewards for each robot. Moreover, both policies were calculated with a C++ implementation of Sarsop in an Intel Core 2 Duo processor @2.47GHz and 2.9GB, being the former computed for 0.2 seconds and the latter for 4677 seconds (the first time is so short because the algorithm converged quite quickly). Then, three different approaches were tested: (i) the fully cooperative policy; (ii) independent policies for each robot but using DDF; (iii) independent policies for each robot without sharing any information. For each simulation, the expected discounted reward was calculated for the whole team. Figure 5.1 depicts histograms for the average rewards obtained during 500 simulations of 150 steps each one.

In this case, with much less computation time, the quality of the proposed coordinated approach was quite close to the fully cooperative. Moreover, even though the computation of the single-robot policy converged quickly within 0.2 seconds, the algorithm kept improving that policy more slowly if run longer. Actually, computing the single-robot policy for 20 seconds the proposed approach even outperformed the fully cooperative that had been computed for 4677 seconds. This shows the huge difference between the two models in terms of complexity for this particular example, and how the coordinated solution can reach sub-optimal results. In general, the reward is usually higher when sharing information compared to the case of no data fusion. Nonetheless, due to the small size of the map, differences were not very substantial in this example. Finally, it is crucial to mention that the map was chosen so small in order to compare with the optimal solution, since the fully cooperative MPOMDP became intractable with these computation resources otherwise.

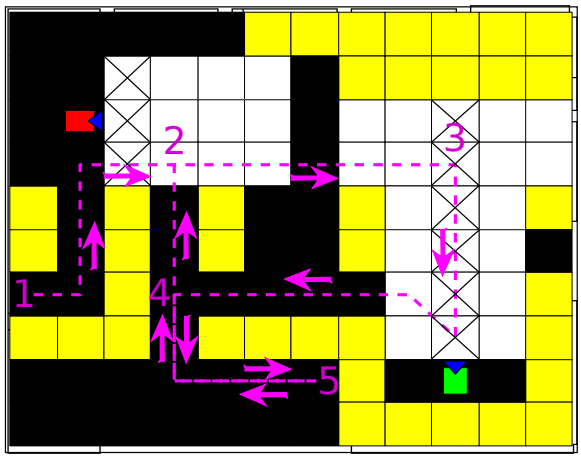


Figure 5.2: Simulated environment and field of view (white cells) for each robot. Cells representing obstacles are in yellow. If the target is in one of the cells with crosses, a high reward is obtained. The path followed by the target is also shown.

5.5.2 Simulations

Additional simulations for more complex environments were conducted in Matlab in order to show the advantages of introducing *mixed observability* and coordination through DDF. The simulated environment was a grid of 2x2-meter cells that resulted in the occupancy map of 12x10 dimensions shown in Figure 5.2, where cells representing obstacles are in yellow.

Again, a team with two robots is considered. Nevertheless, their perception capabilities are different. The first robot carries a more accurate sensor ($p_D = 0.9$) but its FOV is smaller, whereas the second has a wider FOV but is less accurate ($p_D = 0.8$) (see Figure 5.2). That is reasonable, because many vision-based detectors work more accurately when the FOV of the scene is more restricted. Therefore, the role of the robot with the wider FOV would be to survey a big area from a distant position whereas the robot with the more accurate sensor would try to get closer to confirm the target detection. Hence, the first robot gets a high reward (+100) when the target is in one of the closest cells of its FOV, but the second robot gets a high reward (+100) when the target is in one of the central cells of its FOV (see Figure 5.2); otherwise the reward is zero. The discount factor is $\gamma = 0.9$.

	Coordinated	**Not coordinated**
MO-Symbolic Perseus	246.13 ± 3.33	209.38 ± 3.17
Sarsop	279.64 ± 4.67	197.76 ± 3.10

Table 5.1: Average rewards (with 95% confidence interval) obtained with and without coordination (DDF) during the simulations. The value at each simulation is obtained by computing the expected discounted reward for the whole team.

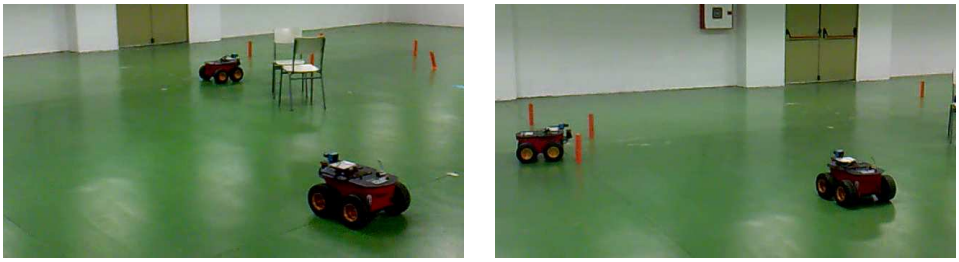
First, the decentralized approach with coordinated MOMDPs proposed in Section 5.4 was tested for this scenario. Independent policies were calculated for each robot with Sarsop and MO-Symbolic Perseus. Then, 500 simulations of 150 steps each one were performed with the robots starting at random positions and the target following the fixed path depicted in Figure 5.2, which was unknown for the pursuers. In order to include some uncertainty in its behavior, at every time step, the target could (with equal probability) either stay in the same cell or follow the path.

Table 5.1 shows the average rewards obtained with and without coordination (DDF) during all the simulations. The value at each simulation is obtained by computing the expected discounted reward for the whole team. It can be seen how, for both solvers, the use of DDF produces an increase of the average reward. Again, it is also crucial to remark that a comparison with the fully cooperative MOMDP is not included here due to the complexity of the domain. Both solvers run out of memory (with the same computer mentioned in the previous Section) trying to compute a fully cooperative policy.

Finally, to show the advantage of introducing the *mixed observability* in the model, some simulations (100 runs of 150 steps) with a single pursuer robot were carried out varying the size of the scenario and comparing with a standard POMDP solution. This robot has the same sensor configuration as the red robot in Figure 5.2 and the discount factor is $\gamma = 0.9$ again. As the size of the scenario is increased, so is the number of possible states for the robot and the target, as well as the complexity of the model to solve. In particular, three different scenarios are considered: a 3x4 grid without occupied cells; the scenario in Figure 5.2 (80 cells); and a 12x10 grid without occupied cells. The results of the average rewards for both, POMDP and MOMDP policies, are summarized in Table 5.2. The values represent an average on

Map size	Type	Time (s)	Precision	Reward
12 cells	MOMDP	0.18	21.97	125.87 ± 6.19
	POMDP	1517	21.99	88.74 ± 8.90
80 cells	MOMDP	1054	54.89	64.12 ± 8.61
	POMDP	3169	54.80	44.49 ± 8.70
120 cells	MOMDP	3071	57.42	42.58 ± 5.84
	POMDP	3081	47.30	40.24 ± 7.18

Table 5.2: Evaluation of POMDP and MOMDP policies for a single robot (rewards with their 95% confidence interval) as the state space is increased. Expected discounted rewards ($\gamma = 0.9$) are considered to compute the average.



(a) (b)

Figure 5.3: Images of the experiments in the real CONET testbed.

the expected discounted reward for each run. All the policies were calculated with the same solver (Sarsop) and the same computer (the one mentioned above), and their precision measured. This precision (Kurniawati et al., 2008) is a variable that Sarsop uses to check convergence, and it represents the gap (average distance) between the upper and lower bounds on the value function computed by the algorithm. For the smallest and medium scenarios, policies were calculated until they achieved a similar precision, being the computation time much shorter and the average reward better for the MOMDP cases. In the biggest scenario, the stop criterion for the policies was the computation time. Thus, this example tries to show how the results can be better for the MOMDP after similar computation time.

5.5.3 Real experiments

In order to show the applicability of the approach, some real experiments were conducted. In these experiments, two robots (Pioneer 3-AT[1]) were used to follow a target represented by a third robot (see Figure 5.3). The experiments were carried out in the multi-robot testbed of the Cooperating Objects Network of Excellence (CONET)[2]. The testbed was configured so that the occupancy map was the same as in the previous simulations (see Figure 5.2). The models considered for the robots were also the same ones.

The software modules used by the robots are shown in Figure 5.4. Player (Gerkey et al., 2003) was used to control the robots, which were able to localize themselves within the map with a Monte-Carlo localization algorithm (AMCL) (Dellaert et al., 1999). Besides, a path planning algorithm (Wavefront) was used to obtain the path to the high level goals provided by the POMDP controller (next cell to move), whereas a local navigation algorithm (VFH) (Ulrich and Borenstein, 1998) was used to safely navigate the given path. More details about the implementations of these algorithms can be seen in the Player/Stage project[3]. Moreover, in order to implement the approach in Section 5.3, each robot had a DDF estimation filter and a POMDP controller that executed the obtained policies. The sample times were 5 seconds for the DDF filter and 10 for the POMDP controller.

Results of the experiment with coordinated execution are summarized in Figure 5.5. The localization of the target is improved, as it can be seen in screenshot 5.5(a), where the belief state of the blue robot is narrow even though the target is not in its FOV. Moreover, a certain coordination arises between the robots. For instance, in screenshots 5.5(c), 5.5(d), 5.5(e), the target escapes and the blue robot goes through its lowest cost path while the red robot keeps observing the place, as there is a certain probability of the target coming back and it has a larger FOV. When most of the probability mass is in the lower left room 5.5(f), both robots eventually move to that direction 5.5(g), and the blue robot moves inside the room while the red waits

[1]These robots are provided by the manufacturer Mobile Robots, http://www.mobilerobots.com.
[2]http://www.cooperating-objects.org.
[3]http://playerstage.sourceforge.net.

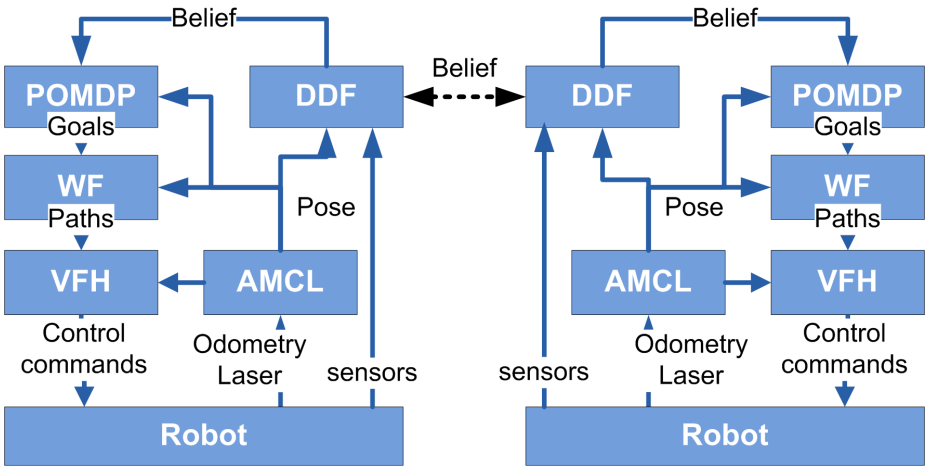


Figure 5.4: Software architecture of the real robots.

observing from afar 5.5(h). Figure 5.6 compares the trajectories of the robots and the target in two different experiments, with and without coordination.

5.6 Conclusions

The poor scalability of POMDP models is a concern for their application to multi-robot planning. This Chapter proposes decentralized data fusion as a manner to coordinate independent plans computed by local POMDPs. This approach using decentralized data fusion is highly scalable because the robots in a team do not need to reason about the actions performed by the other team members. The main drawback is that sub-optimal policies are obtained, and no intentional cooperation is considered. However, a proper design of the local rewards enables the method to cope with different applications using a coordinated team. Moreover, this coordinated decentralized solution was successfully compared to a fully cooperative solution (optimal) by means of simulations.

Besides, the Chapter contributes showing the computational advantages of introducing MOMDPs. Point-based POMDP solvers are adapted to cope with mixed observable states. Real experiments are also included to demonstrate the feasibility of the approach.

The lack of knowledge about others' actions may lead to many situations in which the robots do not exploit their resources optimally. The following Chapter addresses this issue and proposes another approach in which cooperation among the robots is considered explicitly. The challenge is being able to incorporate some information about other team members' actions into the scheme, while maintaining the scalability of the model.

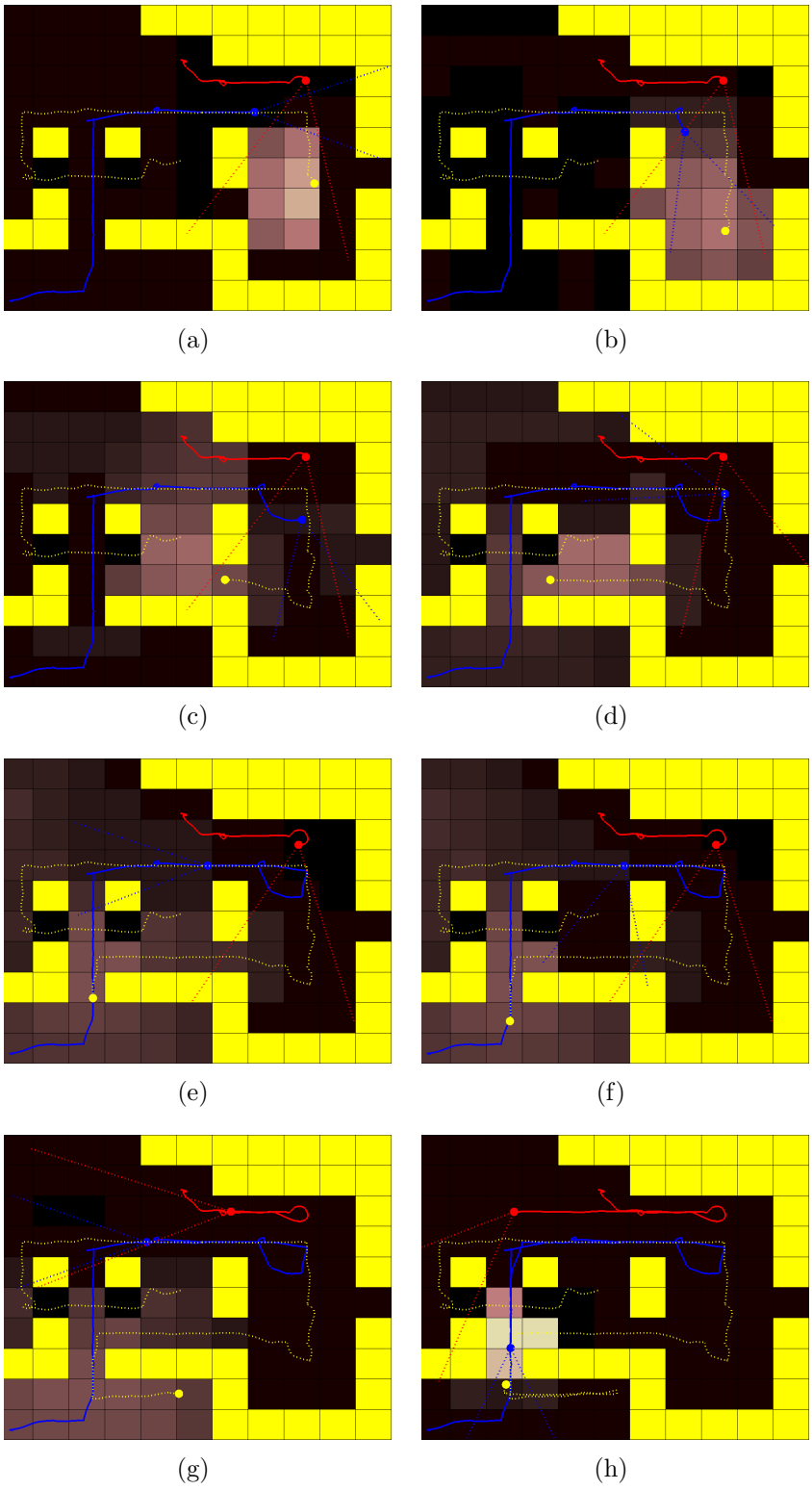


Figure 5.5: Screenshots of a real experiment using coordinated robots. The color scale of the cells represents the (fused) belief from the blue robot. The brighter, the higher the probability. In yellow the target. The headings of the robots are also represented.

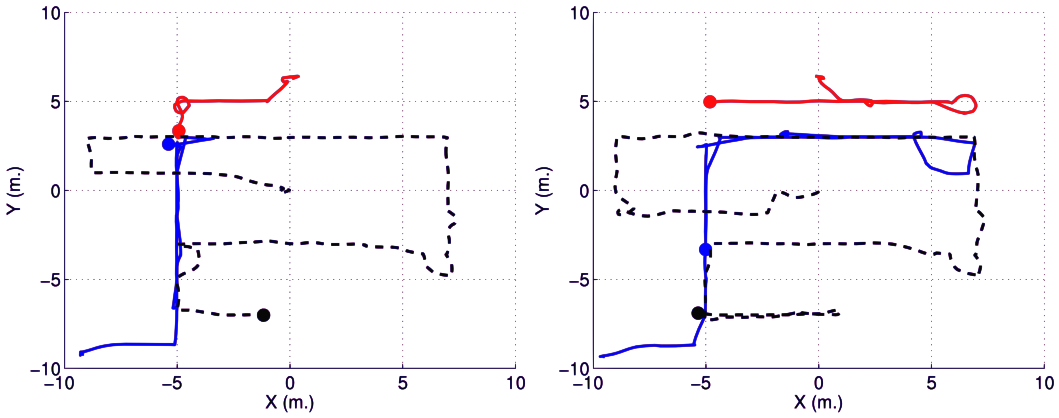


Figure 5.6: Paths of the target (black) and robots. Left: no coordination. Right: coordination. In the first case the robots lost the target in part of the scenario.

Chapter 6

Decentralized cooperation with auctioned POMDPs

Like the previous one, this Chapter addresses the scalability problem that POMDPs face when considering multi-robot teams. In particular, the Chapter proposes to decentralize multi-agent POMDPs by separating the overall goal into several individual behaviors that can be modeled by simpler POMDPs. These behaviors can be represented by single-agent policies and assigned to different robots during the execution of the mission. The objective is to achieve cooperation by distributing these roles among the teammates, but using assignment methods with low communication requirements.

Additionally, communication models in the multi-agent POMDP literature severely mismatch with real inter-robot communication. Here, this issue is again addressed by applying the decentralized data fusion method that efficiently maintains a joint belief state among the robots. This way, the coordinated approach in Chapter 5 is improved by taking a further cooperative step since different behaviors can be distributed among the robots in an efficient manner.

This Chapter also focuses on the same cooperative tracking application described in Chapter 5, in which several robots have to track a moving target of interest jointly. The proposed ideas are illustrated in real multi-robot experiments, showcasing the flexible and robust coordination that this Chapter's techniques can provide.

6.1 Introduction

The methods of this Thesis are aimed at robotic applications in real scenarios in which multi-robot systems are of great interest, such as surveillance or rescue robotics (Merino et al., 2006; Hsieh et al., 2007). In these real environments, robots may experience communication constraints regarding connectivity, bandwidth and delays; and mapping the overall task into robust plans for each robot is a challenging problem. Besides, the high complexity of the general MPOMDP model was already stated in Chapter 4, where it was shown that such model is hardly scalable in line with the number of robots.

In this Chapter, a scheme for achieving cooperation within the team is proposed, while mitigating complexity of the MPOMDP by lowering the dependence of individual plans. Moreover, robotic teams are commonly capable of communicating internally, which will be exploited to maintain a decentralized state estimate. In Section 4.3.8 of Chapter 4, some models in the literature on multi-agent planning under uncertainty (Pynadath and Tambe, 2002; Nair et al., 2004; Roth et al., 2005) were explored. A key point in this Thesis' approaches however, is that there is a relaxation of the strict assumptions on the quality of the communication channel commonly found in those models. In particular, the coordinated approach in Chapter 5 is improved in this Chapter by adding explicit cooperation.

As it has been stated, a fully decentralized solution is considered in this Thesis. That is a solution that involves local information and local communications only, and which does not depend on the total number of robots. In particular, this Chapter presents an intermediate approach that solves independent POMDPs for each robot while still achieving online cooperation during the execution phase by distributing the individual tasks using auctions. Auction algorithms have been widely used for optimal multi-robot task allocation (Mosteo and Montano, 2007; Viguria et al., 2008), and have also been explored in conjunction with POMDPs (Spaan et al., 2010).

This Chapter proposes to decompose a MPOMDP by factorizing its goal into several behaviors that can be represented by single-agent POMDPs. A centralized POMDP auction (Spaan et al., 2010) is generalized to auction never-ending tasks

(behaviors) that can be assigned to different robots at every step without completion (unlike classic task allocation). In this novel decentralized auction, policies that describe a behavior toward a common goal are allocated instead of tasks; robots can switch between these behaviors dynamically at each decision step, and the auction is used to continuously determine which behavior suits each robot best to attain the goal cooperatively. Since independent single-agent POMDPs are solved for every single robot, the interdependence of the model is low and the approach can scale well in line with the number of robots.

The second key component is to efficiently maintain a joint belief state among the robots, which can serve for coordination. For this purpose, a DDF approach in conjunction with POMDP policies is used, as it was done in Chapter 5. Unlike most works on POMDPs, the belief update here is separated from the decision-making process during the execution phase, allowing the agents to estimate the state with the DDF algorithm. This decoupling between both processes increases the robustness and reliability of real-time robotic teams.

Once again the method is illustrated through a multi-robot tracking application, in which several robots have to cooperate in order to track a moving target as accurately as possible (active perception). In any case, these techniques also suit a wider range of problems, such as surveillance (Hsieh et al., 2007) or forest fire detection (Merino et al., 2006), which call for a cooperative effort of robots coordinating their individual behaviors. In particular, the proposed distributed POMDP approach is demonstrated in a multi-robot testbed with a fully decentralized setup where each robot runs its own POMDP policy.

6.2 Distributed MPOMDP

In Chapter 4, the two main sources of complexity of the MPOMDP model were analyzed: complexities due to the high dimension of the belief spaces; and due to the inclusion of multiple agents, which makes the model non-scalable. The high-dimensional belief spaces were alleviated in Chapter 5 by assuming mixed observability within the

model. This Chapter also uses MOMDPs in order to tackle this issue, allowing the approach to cope with domains of real applications.

On the other hand, the increase of complexity caused by adding new agents was addressed in Chapter 5 by separating the complete model into single-agent models, and arousing an implicit coordination during the execution phase thanks to a decentralized data fusion approach. In this Chapter a similar decomposition is proposed, but a further step is taken, because robots cooperate explicitly during the execution of the mission by exchanging different roles or behaviors.

There is a wide range of missions that can be accomplished by a team of multiple robots. In all these missions there is a certain objective (e.g., detecting a target or alarm) and a set of behaviors or roles that the robots can follow to achieve that objective (e.g., patrol, approach, etc.). In the case of a MPOMDP, this overall objective is encoded into a single reward function. As it was stated in Chapter 5, without losing generality, in many applications this reward function can be seen as an addition of a series of reward functions for the different robots:

$$R(s, a^J) = R^1(s, a^J) + \cdots + R^N(s, a^J) \tag{6.1}$$

All these reward functions should be defined over the joint actions and a joint belief which considers information about the whole team, i.e. observations and actions from all the robots. Thus, a MPOMDP can find an optimal solution that reasons about all these actions and observations.

In addition, here it is assumed that the general mission can be decomposed into different simultaneous behaviors, each of which is modeled as a POMDP with its own reward function. Then, the joint behavior produced by the distributed POMDPs should be similar to the one desired for the whole team initially. Such a decomposition is possible in many robotic applications (Merino et al., 2006; Hsieh et al., 2007), such as surveillance, tracking, forest fire detection or robotic soccer, in which cooperation between robots playing different roles is required.

Therefore, the global multi-agent objective can also be distributed into a set of simpler reward functions $\{R_1, \ldots, R_M\}$. Each reward function R_k represents a certain single-agent behavior that can be modeled by a POMDP policy with a value function $V_k^{\pi}(b)$ associated. Then, assuming that each of the robots can execute any of these policies, the combination of all of them should lead to a cooperative behavior that follows the global objective. The problem of determining online which policy should be assigned to each robot at each step can be modeled as a task allocation problem (Spaan et al., 2010). Hence, the reward functions in (6.1) for each robot should match one of the M reward functions that describe the possible behaviors.

There are M single-agent POMDPs (one for each of the possible behaviors to assign) that can be solved for each robot considering only their local information (action and observation). However, once the policies have been obtained, they are executed over a joint belief which encodes information from all the robots. This joint belief is determined with the same decentralized data fusion approach explained in Chapter 5. In that previous Chapter, it was also highlighted that the optimal solution can only be obtained if the POMDP policies are solved considering joint information instead of local. Nevertheless, the approach proposed here is a compromise to obtain a sub-optimal solution that is scalable in line with the number of robots. Moreover, during the execution phase, unlike in the approach in Chapter 5, the robots cooperate explicitly, since the different policies are allocated to different robots continuously so that they coordinate their behaviors. Of course, in order to keep scalability, these assignments should be done by a decentralized method that does not require remarkable communication constraints.

6.3 Decentralized auction with POMDPs

As mentioned above, this Chapter focuses on decentralized models. Recall that throughout this Thesis, a decentralized system has been defined as one in which (Nettleton et al., 2003):

1. There is no central entity required for the operation.

2. There is no common communication facility, that is, information cannot be broadcast to the whole team, and only local point-to-point communications between neighbors are considered.

3. The members do not have a global knowledge about the team topology: they only know about their local neighbors.

These characteristics make the system scalable as it does not require a central node and enough bandwidth to transmit all of the information to that node. Moreover, the system is more robust and flexible with respect to loss or inclusion of new agents (there is no need to know the global topology), and with respect to communication issues (a failure does not compromise the whole system).

The approach proposed in this Chapter builds on two mechanisms for achieving decentralization: a decentralized data fusion filter (see Section 6.3.1) for sharing information between agents recovering a common belief; and a POMDP auction for decentralized behavior cooperation (see Section 6.3.2). In terms of agent interdependence, this approach can be seen as in between "independent POMDPs" and MPOMDP/Dec-POMDP within the classification done in Section 4.3.8 of Chapter 4. In terms of communication requirements, it does not require the high-quality guarantees of the methods that enhance the Dec-POMDP model with communications.

6.3.1 Decentralized data fusion

Given a team with N agents, a centralized node with access to all of the information would update the belief according to:

$$b_{cen}(s') = \eta p(z^J|a^J, s') \sum_{s \in S} p(s'|a^J, s) b_{cen}(s) \tag{6.2}$$

However, if the system is decentralized and each agent i only uses its local information (action a^i and observation z^i), this centralized belief could still be recovered locally by each agent, as it was explained in Chapter 5. The same DDF scheme that was used in that Chapter is applied here. Thus, it was shown how, under certain

circumstances, if the agents exchanged their beliefs, these could be combined in the following manner:

$$b_{cen}(s') \leftarrow b^i(s') \prod_{j \neq i} \frac{b^j(s')}{b^{ij}(s')} \tag{6.3}$$

where $b^{ij}(s')$ represented the common information between the agents i and j, that was maintained by the so-called Channel Filter.

In a realistic scenario, each agent will be in communication range only with a subset of the other agents, and (6.3) is applied only considering this subset. However, as each agent locally accumulates information that shares with its neighbors, the belief will propagate through the network, allowing the agents to obtain a common belief. The Channel Filters assume that there are no loops in the information channels, that is, the network topology is a tree. Otherwise, the problem of double-counting of information (rumor propagation) should be taken into account as well by means of conservative fusion rules.

6.3.2 Distributed auction of POMDP policies

The robots can cooperate by switching from certain behaviors to others. Each behavior is represented by a policy that the robot has to follow, and the problem of deciding in some optimal manner which policy should be assigned to each robot at every moment can be formulated as a task allocation problem.

In general, a task allocation algorithm attempts to assign a set of M tasks to a team of N agents minimizing a global cost. In this case, each robot always has to be assigned a sole task, which is the POMDP policy to follow. In order to foster cooperation, different policies must be assigned to different robots as long as possible. Provided that a variable $x_{ik} = 1$ when policy k is assigned to robot i and 0 otherwise, and c_{ik} is the cost associated with that assignment, the problem can be formulated as follows:

$$\min \sum_{i=1}^{N} \left(\sum_{k=1}^{M} c_{ik} x_{ik} \right)$$

subject to

$$\begin{aligned} &\sum_{i=1}^{N} x_{ik} \leq 1, \quad \forall k \in \mathcal{K} \\ &\sum_{k=1}^{M} x_{ik} = 1, \quad \forall i \in \mathcal{I} \\ &x_{ik} \in \{0,1\}, \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K} \end{aligned} \tag{6.4}$$

where $\mathcal{I} = \{1, \ldots, N\}$ and $\mathcal{K} = \{1, \ldots, M\}$.

In order to select the best behavior for each robot, an auction algorithm similar to the one presented by Spaan et al. (2010) is proposed. The cost or bid of assigning a policy k to a robot i is $c_{ik} = -V_k^{\pi}(b_i)$. Thus, policies with a greater expected reward are more likely to be selected for each robot, which helps to maximize the global expected reward for the whole team. This is an advantage of using an auction algorithm, since the knowledge obtained by the POMDP policies can be exploited. Moreover, auctions can be performed in a decentralized fashion by exchanging only the bid values, so no remarkable communication load is added to the system.

Algorithm 9 summarizes this decentralized auction approach in which the assignment problem is solved at each robot locally with the information available. Each robot i computes its own bids for the behaviors from its local belief b^i and communicates them to other neighboring robots (lines 2 and 3). Then, with the bids received from other robots, a local solution for the assignment problem in (6.4) is obtained (line 7). All the robots become bidders and auctioneers, since they bid for policies and solve the assignment problem at the same time. Furthermore, the cost matrix obtained for a robot i that has received bids from $j-1$ other robots ($j \leq N$) is calculated in line 6 and it would be:

Algorithm 9 Auctioneer robot i (b^i)

1: **for all** $k \in \mathcal{K}$ **do**
2: $c_{ik} = -V_k^{\pi}(b^i)$ {; Local bids}
3: Send c_{ik} to neighbors.
4: **end for**
5: Receive bids from neighbors.
6: $C = \{c_{ik}\}_{i,k}$ {; Create cost matrix}
7: $\{x_{ik}\}_{i,k} \leftarrow Hungarian(C)$
8: **return** Policy selected for robot i.

$$C = \begin{pmatrix} -V_1^{\pi}(b^1) & \cdots & -V_M^{\pi}(b^1) \\ \vdots & & \vdots \\ -V_1^{\pi}(b^i) & \cdots & -V_M^{\pi}(b^i) \\ \vdots & & \vdots \\ -V_1^{\pi}(b^j) & \cdots & -V_M^{\pi}(b^j) \end{pmatrix} \tag{6.5}$$

In order to solve the assignment problem in polynomial time, the Hungarian algorithm (Burkard, 2002) is used. Given the cost matrix C, where each row represents a robot and each column a policy, this algorithm can find a matching between policies and robots so that the total cost is minimized. The complete procedure is described in Algorithm 10.

It important to remark that in case $N > M$, the allocation algorithm will leave robots with no policy assigned. Therefore, the assignment problem is repeated with these free robots (and the whole set of policies) until they all get a policy assigned. Note that in this special case, some policies would be assigned to more than one robot at the same time, but this is better than not using those robots at all. In other applications, those robots may stay idle.

In general, the local cost matrices (6.5), and hence the local solutions for the behavior assignment, should be the same at each robot as long as the communication is error-free and the beliefs are common (thank to the DDF). However, for DDF systems in which the local beliefs are not synchronized all the time, inconsistencies that lead to sub-optimal solutions may be obtained from time to time. In an inconsistent

Algorithm 10 $\{x_{ik}\}_{i,k} \leftarrow \text{Hungarian}(C)$

Step 1: Ensure that the matrix C is square by adding dummy rows/columns if necessary. Conventionally, each element in the dummy row/column is the same as the largest number in the matrix.
Step 2: Reduce the rows by subtracting the minimum value of each row from that row.
Step 3: Reduce the columns by subtracting the minimum value of each column from that column.
Step 4: Cover the zero elements in the matrix with the minimum number of lines possible. If the number of lines is equal to the number of rows then go to Step 8.
Step 5: Add the minimum uncovered element (uncovered means with no line on it) to every covered element. If an element is covered twice, add the minimum element to it twice.
Step 6: Subtract the minimum element from every element in the matrix.
Step 7: Cover the zero elements again. If the number of lines covering the zero elements is not equal to the number of rows, return to Step 5.
Step 8: Select a matching by choosing a set of zeros so that each row or column has only one selected.
Step 9: Apply the matching $\{x_{ik}\}_{i,k}$ to the original matrix, disregarding dummy rows.
return $\{x_{ik}\}_{i,k}$

distributed solution, due to differences in the local cost matrices, the same policy is allocated to more than one robot. Therefore, a good synchronization of the local beliefs is desirable to avoid these situations. Nonetheless, the robustness of the system is high, since information from all the robots is not required to compute each local solution. In case some communication links failed, each robot would still get a sub-optimal solution with the available information from their neighbors (sub-networks arise naturally).

In addition, there is another potential desynchronization due to the fact that the robots may not have synchronized execution time, which would lead them to make decisions at different moments and with different available information. Some previous works (Choi et al., 2009) propose consensus algorithms over this information in order to guarantee convergence for decentralized auction approaches even in case of time desynchronization. Nonetheless, the decision-making performance is still degraded.

Moreover, the dynamics of the systems presented here are higher, since each robot is allowed to change its policy at every step. Therefore, the establishment of a previous consensus to converge to the same distributed solution is not worthwhile.

6.3.3 System overview

The system elements per robot are illustrated in Figure 6.1. The whole process is separated into two different modules. Each robot can execute a certain number of behaviors modeled as single-agent POMDP controllers. A DDF module is in charge of computing the belief and feeding the Auctioneer module, which then chooses the adequate POMDP controller and the associated action.

Even though most POMDP-based systems synchronize belief update and decision-making in the same loop, here both processes are separated. Thus, some constraints that limit the flexibility and robustness of the system are avoided. For instance, communication channels and transmission rates are totally independent for both modules, which is critical in decentralized systems under possible communication failures. Nonetheless, it is worth mentioning that this choice has also some theoretical implications. Basically, the beliefs provided by the parallel DDF system to the POMDP could be different from the ones that were used during the computation of the policy, where the belief update was integrated into the POMDP. Anyway, in the case of point-based POMDP solvers, the error that is made by evaluating the value function in a belief point which does not belong to the initial belief set (the one used to compute the policy) is bounded (Pineau et al., 2006).

The approach is totally decentralized, since the belief estimation as well as the decision-making are carried out without the need for a central entity. On the one hand, the belief estimation is computed by a DDF algorithm that is distributed along the multiple robots. On the other hand, the POMDP controllers also act for each robot separately. Despite the fact that a multi-agent POMDP for the whole team is not solved (with its computational benefits), a cooperative behavior still arises in two manners. First, thanks to the information shared by the different DDF modules in order to achieve a fused belief (which then acts as a Markovian coordination signal for

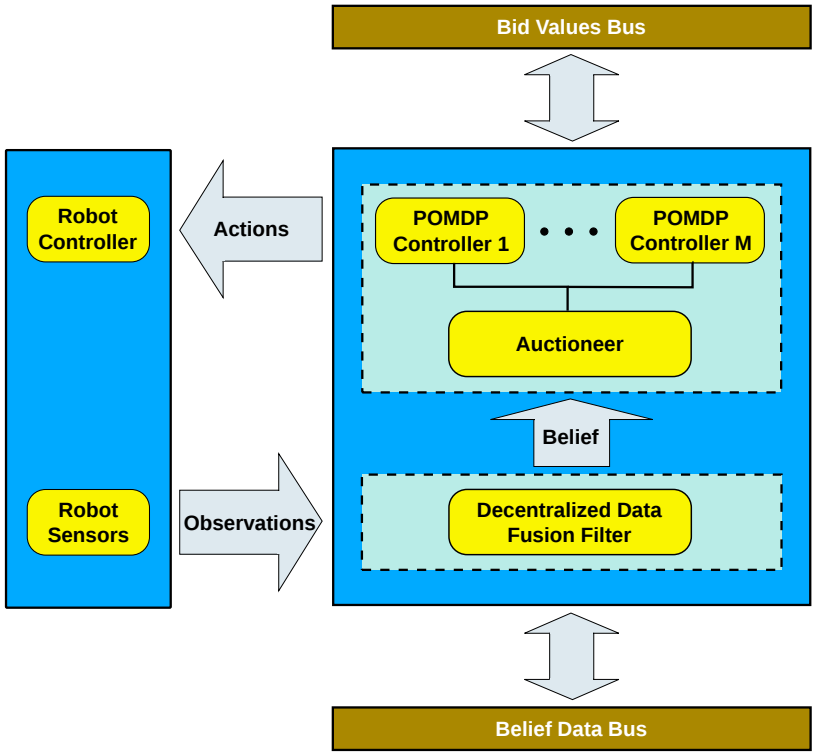


Figure 6.1: Functional scheme for decision-making and belief update at each robot.

policy execution); and second, by sharing the bid values for the decentralized auction, what gives an idea about the behaviors others may be performing.

6.4 Multi-robot cooperative tracking

In order to illustrate the proposed approach, the same application that was presented in Chapter 5 for tracking a target by means of multiple robots is considered here. In this problem there was a moving target to track, and a team of N pursuer robots with bearing-only sensors.

As a summary, recall that the state for each robot was composed of the position of the target and its own position and heading. The state space was discretized into a cell grid, and a map of the scenario was assumed to be known. There were four possible headings for every robot: *north*, *west*, *south* or *east*; four possible actions:

stay, *turn right*, *turn left* or *go forward*; and a boolean measurement: *detected* or *non-detected*. Moreover, each sensor was defined by a FOV and a probability of detection p_D.

The design of the reward function is crucial in order to achieve active perception by reducing the uncertainty in the target position. Since the target must be tracked by the team, the more robots have it within their FOV, the higher reward the system should be given. Besides, given that the sensors provide bearing information, it is quite reasonable to reward cross configurations between the robots. Bearing sensors entail mainly uncertainty in depth, so pointing at the target from different angles definitely helps to reduce the uncertainty of its estimation. Therefore, a high reward should be given for each robot that is keeping the target within its FOV, and even higher if the robot's orientation differs from the others'.

A multi-agent POMDP might be solved in order to deal with this problem. However, this solution is far from scalable in line with the number of robots. Actually, even considering just two robots and a moderate number of cells for the grid (~ 80), the problem becomes intractable (considering the solvers and the computers indicated in the experimental Sections of this Thesis). Hence, the method presented in this Chapter to distribute the reward function is used.

The key idea is to distribute the reward function so that the same desired behavior is obtained for the whole team. In this case, the robots should track the target from different directions, so the kinds of behaviors to be allocated could consist of following the target from a specific direction. For this application, four single-robot behaviors are considered, one for each possible orientation $\{north, west, south, east\}$. The reward function for the policy k (R_k) gives a high reward to robot i only if the target is within its FOV and the robot's heading rob_h^i corresponds to the orientation of behavior k. Since the objective is to track the target, the robots should try to get closer to the target when possible. Hence, the high reward is just obtained when the target is in one of the closest cells. In this case, homogeneous robots with similar capabilities are considered. Therefore, the FOV for all the robots and the corresponding cells with a high reward are represented in Figure 6.2.

Although a cooperative tracking application is presented here, it is important to remark that the framework is more general. Actually, it can suit many other multi-robot applications in which different behaviors should be combined in a cooperative manner. For instance, in robotic soccer the teammates have to keep switching roles continuously depending on the situation; in surveillance applications the robots have to distribute themselves to patrol different areas; in fire-fighting applications or rescue robotics, heterogeneous robots can play different roles during a mission according to their capabilities.

Finally, in order to alleviate the complexity of the belief space, MOMDPs are considered to find the policies as it was done in Chapter 5. Hence, the robots' states are assumed to be observable within the POMDP. This is reasonable for this robotic task in which the on-board sensors allow the robots to assume their own states observable (at least for a given resolution).

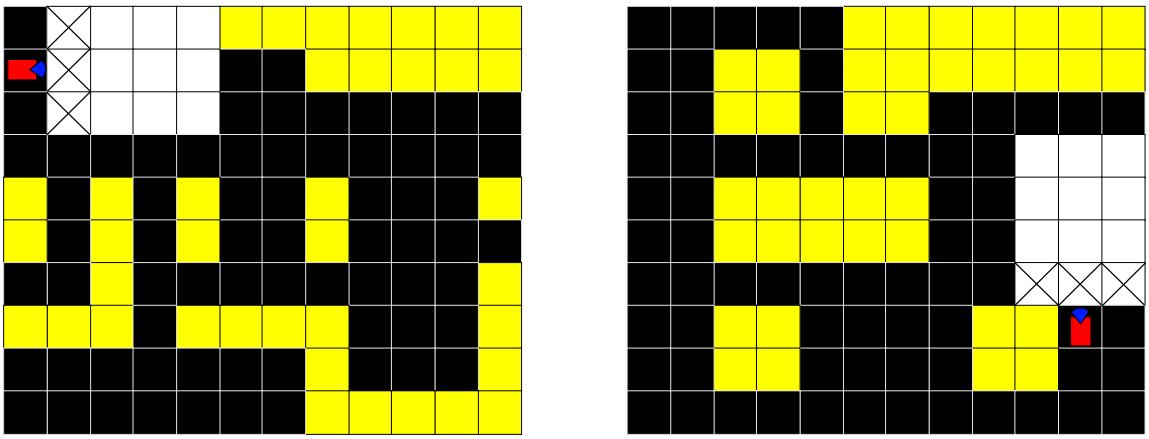
6.5 Experimental results

Some experiments were conducted with the real multi-robot testbed that was created by the Cooperating Objects Network of Excellence (CONET) [1], the same one that was used in the real experiments of Chapter 5. The testbed is composed of four real robots (Pioneer-3AT) in an indoor facility as well as a simulated version in Player/Stage (Gerkey et al., 2003). Besides, there is the possibility of combining real robots with simulated robots, which enables the user to experiment with a higher number of robots. Figure 6.3 shows different views of the multi-robot testbed.

6.5.1 Experimental setup

The size of the experimental area for the robots was 24×20 meters, and there were moving walls that could be relocated in order to obtain different map configurations. The testbed area was discretized into 2×2-meter cells and occupancy grids were obtained for the different map configurations. A cell was considered to be occupied

[1] http://www.cooperating-objects.org.



(a) (b)

Figure 6.2: (a) and (b) are the occupancy grids of the testbed for two different map configurations. Yellow cells are obstacles, whereas the remaining ones are accessible. An example of the FOV for a robot is shown in white cells. All the robots have the same FOV. Besides, if the target is in one of the cells with crosses and the heading is adequate, a high reward is obtained.

whether there was an obstacle on it or it was unattainable for the robots. Figure 6.2 shows the occupancy grids for the two map configurations that were used during the experiments of this Chapter.

A team of robots was considered in order to follow a target, represented by another robot. All the pursuers presented similar perception capabilities by means of a sensor with $p_D = 0.9$ and the FOV shown in Fig. 6.2. For each POMDP, the high reward when the target was in one of the close cells of the FOV was 100, otherwise the reward was 0. The discount factor was $\gamma = 0.99$. Note that the pursuer observations were obtained by simulating sensors with the mentioned capabilities on board the robots, since the development of real detectors is out of the scope of this Thesis.

During all the experiments the target followed a path unknown for the pursuers and with a random component. Furthermore, Player[2] was used to command the robots, which were able to localize themselves within the map (AMCL algorithm). A path planning algorithm (Wavefront) was used to obtain the path to the high-level

[2]See http://playerstage.sourceforge.net for more details about Player's algorithms.

(a) (b)

Figure 6.3: Pictures of the multi-robot testbed from different points of view.

goals provided by the POMDP controllers (next cell to move and robot heading), whereas a local navigation algorithm (VFH) was used to navigate the given path safely. Each robot had running on-board an estimation filter implementing the DDF scheme in Section 6.3.1 and an auctioneer controller that executed the algorithm in Section 6.3.2.

Three different approaches were tested: (i) auctioned POMDPs with DDF; (ii) auctioned POMDPs without DDF; (iii) independent POMDPs with DDF. The two first approaches are based on the auction method proposed in this Chapter, but in the second one, neither communication nor fusion are considered for the DDF modules. The third approach is the one proposed in Chapter 5, in which a single and independent POMDP is used for each robot and communication between the DDF modules is allowed. Moreover, all the policies were obtained by solving the corresponding MOMDPs with a C++ implementation of the Sarsop algorithm (Ong et al., 2009). The solver ran 1700 seconds for each policy in a computer with an Intel Core 2 Duo processor @2.47GHz and 2.9GB. For the approaches (i) and (ii), a different MOMDP is solved for each heading, whereas for the approach (iii), there is a single MOMDP independent of the heading.

6.5.2 Simulations

First, some simulations to compare the different approaches mentioned above were carried out with 2 robots pursuing another one. These experiments were run in the

		Auction+DDF	**Auction**	**Coordinated**
Target error (m)	Robot 1	5.69 ± 0.13	7.16 ± 0.13	5.41 ± 0.11
	Robot 2	5.68 ± 0.13	8.07 ± 0.14	5.38 ± 0.11
Belief entropy	Robot 1	2.54 ± 0.028	3.42 ± 0.025	2.89 ± 0.026
	Robot 2	2.47 ± 0.030	3.43 ± 0.025	2.89 ± 0.026
Target stopped (%)		72.39	73.26	68.44

Table 6.1: Average results of the simulations with a two-robot team for the different approaches.

simulated version of the testbed, and the configuration of the map was the one shown in Figure 6.2(a). Thus, 5 simulations, of 20 minutes each, for each of the three approaches were performed. In order to have similar conditions for each simulation, all the robots always started at the same fixed points, and presented a sample time of 10 seconds for the decision-making modules and 3 seconds for the DDF modules.

Some average results with their standard deviations are presented in Table 6.1. At each time step, the target localization is estimated by searching the cell of the belief with a highest probability. This value is compared to the actual target position. In general, it can be noticed that the position estimated for the target is improved by using the DDF. Note that, in this case, the resolution of the cells (2 meters) represents a lower bound on the mean errors. The entropies of the belief at each step ($\sum_{\forall cell} -p_{cell} \log(p_{cell})$) are also averaged and shown. Finally, the percentage of time that the target was stuck within the same cell without moving is calculated. It can be seen that, the approach proposed in this Chapter (Auction+DDF) reduces the entropy and increases the target stop time with respect to the Independent-POMDP approach: the cooperation between the members of the team allows them to surround the target, not letting it move away. The option of independent POMDPs without DDF resulted in a very poor performance (and hence it is not included), as the robots do not share any information nor coordinate their tasks.

Due to the bearing information encoded in the sensor observations, a cross configuration among the pursuers allows them to point at the target from different points of view and reduce the uncertainty of their estimations. This cross configuration is fostered by the auctioned approach, as it can be seen in Figure 6.4. This Figure

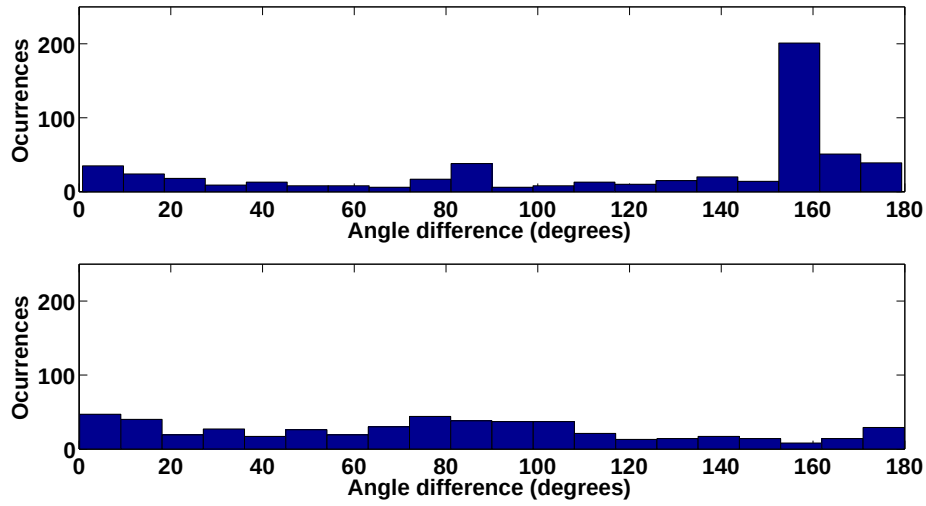


Figure 6.4: Histograms of the angle differences between two simulated robots when the target is within the FOV of any of them. On top Auction+DDF; at the bottom Independent POMDPs+DDF.

compares the auction approach with DDF to the Independent POMDPs with DDF in terms of angle configuration between the pursuers. Histograms of the angle difference between the pursuers every time the target is within FOV are shown. The Auction+DDF histogram presents a high peak around 160° and a small mode in 90° (cross configurations), whereas the histogram is quite flat for the Independent POMDPs. This shows that the proposed approach is more effective in reaching cross configurations.

Second, an experiment to check how the auction algorithm performs when the robots do not access the same information was carried out. The same policies and conditions as in the previous experiment were used. However, the communication rate for the DDF modules was varied throughout the different simulations (two simulations of 20 minutes for each rate value). Even though the communication rate for the auctioneers may also have been varied, the amount of data transmitted by them (bid values) is not significant for the bandwidth when compared to the belief information and, hence, it can be maintained fixed. The performance of the distributed auction algorithm is shown in Figure 6.5 by representing the percentage of inconsistent assignments. Recall that a policy assignment was considered to be inconsistent

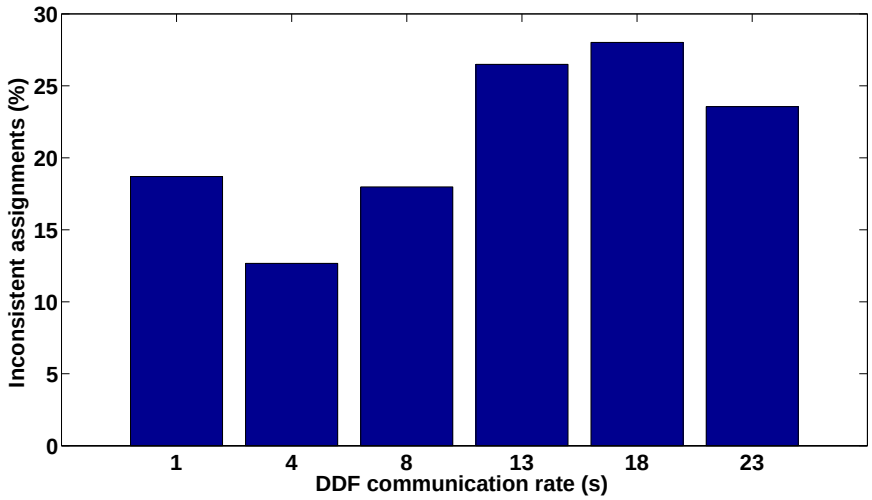


Figure 6.5: Evolution of the performance of the distributed auction under variable transmission rate for the DDFs.

when the same policies are not allocated to the same robots for each local solution of the auction. As the communication rate for the DDFs is increased, the difference between the beliefs to which the agents have access grows too. Thus, the consistency of the assignments becomes more difficult under worse communications. However, Figure 6.5 shows a graceful degradation with respect to the communication rate.

6.5.3 Real experiments

Similar experiments were repeated with the real multi-robot testbed in order to prove the applicability of the methods proposed. Besides, the intention was to use a map where the robots had to face more difficult situations. The configuration in Figure 6.2(b) was used, since there are more possible paths and blind spots that can be exploited by the target.

Again, some experiments were carried out in order to compare the different approaches[3]. Three of the real Pioneer-3AT were used to track the remaining one, that played the target's role. In order to have similar conditions for each experiment, the three robots always started at the same fixed points and the sample times were the

[3]See video at http://vimeo.com/18898325.

	Error(m)	Entropy
	Auction+DDF	
Robot 0	4.07 ± 0.16	2.61 ± 0.05
Robot 1	3.95 ± 0.15	2.55 ± 0.05
Robot 2	4.18 ± 0.16	2.66 ± 0.05
	Independent+DDF	
Robot 0	6.86 ± 0.32	2.80 ± 0.05
Robot 1	6.70 ± 0.32	2.70 ± 0.05
Robot 2	6.75 ± 0.32	2.68 ± 0.05
	Auction	
Robot 0	9.74 ± 0.29	3.85 ± 0.03
Robot 1	9.41 ± 0.34	3.28 ± 0.06
Robot 2	10.46 ± 0.40	3.62 ± 0.04

Table 6.2: Average results of the experiments with a three-robot team for three different approaches. The error of the estimated position of the target with respect to its actual position and the entropy of the estimated beliefs are shown.

same as before: 10 seconds for the decision-making modules and 3 seconds for the DDF modules. An experiment of 15 minutes was performed for each of the three approaches.

Average results with their standard deviations are summarized in Table 6.2. In particular, the target localization errors and the belief entropies are shown for every robot and experiment. The results are similar to the simulated ones. Thus, it can be seen that the Auction+DDF approach reduces the entropy and the target localization error with respect to the Independent POMDPs, since the cooperation between the members of the team allows them to surround the target, pointing at it from different points of view. It can also be noticed that the estimation of the target position is worse for the auctioned approach when no DDF is included. Also in this case, the mean errors are bounded by the resolution of the cells (2 meters), since they cannot be lower.

Figure 6.6 compares the approach of this Chapter to the Independent POMDPs with DDF in terms of angle configuration between the pursuers. Normalized histograms of the maximum angle difference between any of the pursuers every time the target is within FOV are shown. The Auction+DDF histogram presents a high peak

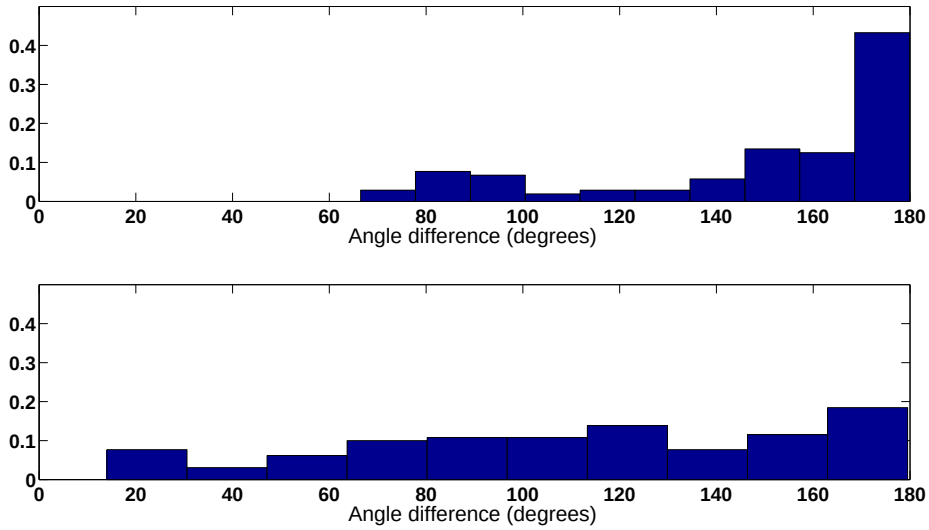


Figure 6.6: Normalized histograms of the maximum angle differences between three real robots when the target is within the FOV of any of them. On top Auction+DDF; at the bottom Independent POMDPs+DDF.

around 180° and a small mode in 90° (cross configurations), whereas the histogram is quite flat for the Independent POMDPs. This shows how the cross configurations are fostered by the auction approach more efficiently.

Second, to show the scalability of the system, a tracking experiment with a four-robot team was performed. In this case, the target was represented by a simulated robot. This experiment was run successfully for more than 30 minutes with the algorithms working on board the robots in a distributed way and using Wi-Fi communications, showing the robustness of the whole system. An extract of the trajectories followed by the pursuers and the target can be seen in Figure 6.7(a). The orientation of the pursuers at the end of the experiment is also plotted to show how they surround the target to reduce the estimation uncertainty. Note that, when the target turns right, since they know the map, the pursuers opt for going directly to the other exit of the aisle so they can find it there.

The cooperation between the members of the team is depicted in Figure 6.7(b), which shows the policies allocated to each robot during the same time frame (each iteration takes place every 10 seconds). Due to differences in the local beliefs and

different decision times for the robots, inconsistent solutions (robots with the same policy) are obtained in some occasions. However, as time passes the robots have built up a better belief regarding the target's position, and the assignment stabilizes (after iteration 22 in Figure 6.7(b)).

6.6 Conclusions

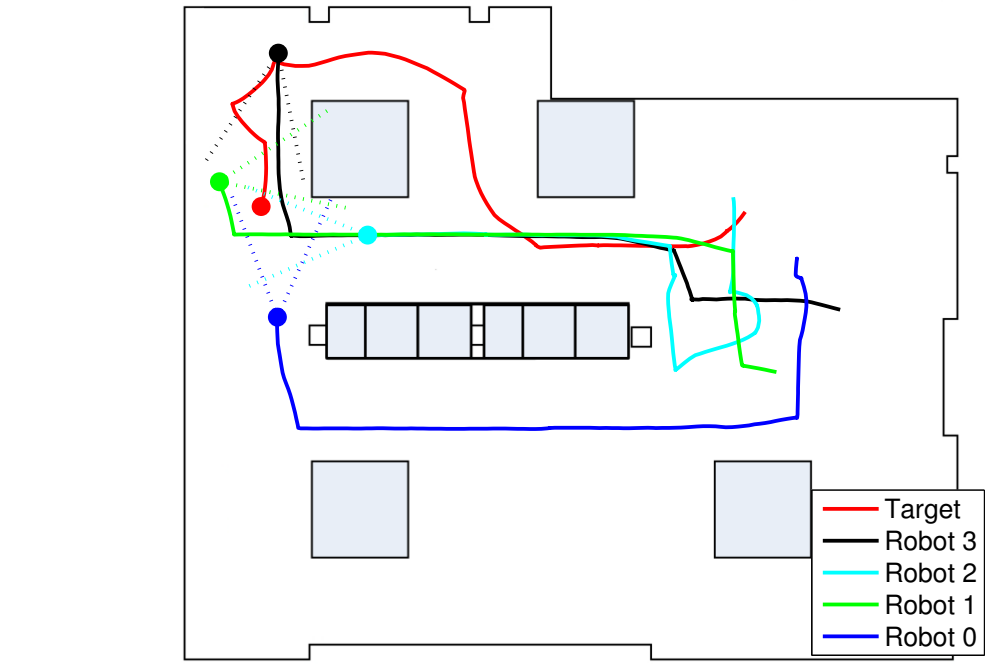
Techniques for planning under uncertainty, such as POMDPs, face a scalability problem when being applied to teams of robots. Popular frameworks like Decentralized POMDPs scale poorly when many agents are considered, unless very severe independence assumptions are imposed (e.g., the ND-POMDP model). Furthermore, many of these models either do not allow agents to exploit inter-agent communication, or implicitly assume instantaneous cost-less communication (MPOMDP). Like Chapter 5, this Chapter focuses on a scalable technique that enables robots to take advantage of available communication channels without requiring strict communication guarantees. Such guarantees are hard to meet in multi-robot domains, typically showing unreliable wireless communications.

Both the approach in Chapter 5 and the one in this Chapter trade off optimality for scalability, leading to sub-optimal solutions of the general MPOMDP that can be applied to real robotic teams. However, whereas the approach in Chapter 5 achieves coordinated behaviors implicitly without reasoning about the actions of the other robots, the approach proposed in this Chapter adds an explicit cooperative layer to the system. This is done by means of the policy auctions, since each robot gets an idea of the behaviors that the others are following at every moment and takes this understanding into account to decide its own behavior.

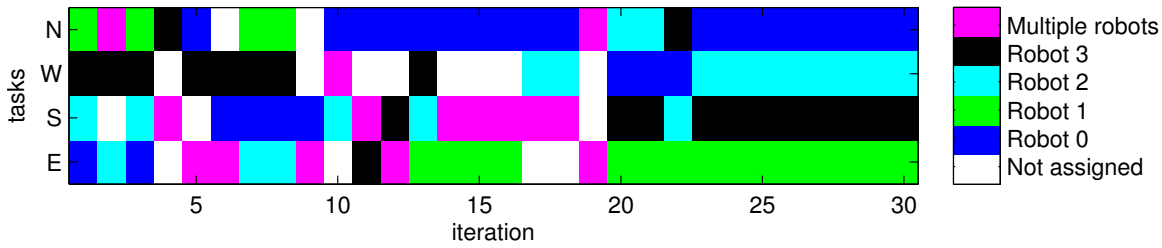
Therefore, the approach presented in this Chapter combines decentralized data fusion with auctions of independent POMDP-based controllers during the execution phase to generate a cooperative behavior in the team. As mentioned before, this approach is much more scalable than other multi-agent POMDP approaches, and enables the robots to exploit imperfect communication channels, thus offering a compromise between optimality and applicability. This Chapter presented experiments

with up to 4 robots, which showed how cooperative behavior develops. This same example cannot be solved with the current state-of-the-art MPOMDP solvers.

Even though a cooperative tracking application is presented as a proof of concept, the framework can be generalized. DDF can be used for policy execution coordination in other multi-robot applications; and any applications that can be achieved through cooperative behaviors can be modeled with this framework. For instance, the method can be used in robotic soccer (allocating the best behaviors/roles to the team depending on the current belief); and in fire-fighting applications (Merino et al., 2006).



(a)



(b)

Figure 6.7: Experiment with a four-robot team. (a) Trajectories followed by the robots and the target during the experiment. Orientations at the last time step are also shown. (b) Policies allocated to each robot during the same time frame.

Chapter 7

Conclusions and future developments

This Chapter summarizes the main contributions of the Thesis and highlights its results. Advantages and disadvantages of the proposed methods are discussed. Furthermore, in order to overcome these drawbacks, some guidelines are given for future extensions.

7.1 Conclusions

The main contribution of the Thesis is to propose a decentralized approach for cooperative active perception between a team of robots, which means that the different individuals take decisions in order to improve the perception of the whole group. This approach is aimed at real robotic platforms, where unreliable communications and real-time constraints can arise.

Due to this focus on applicability, the Thesis proposes decentralized methods that are able to cope with communication delays and breakdowns. This decentralized approach strengthens the robustness of the system, which is a key feature when working in real environments. Since such environments present high variability, probabilistic techniques also become a way of improving the robustness of the systems. Besides, the results of the Thesis were tested for realistic domains involved in real applications,

unlike other existing theoretical multi-robot approaches, for which only academic examples are shown. This is possible because scalability in line with the number of robots is achieved, and all the algorithms proposed can run efficiently under real-time constraints.

The complete problem of active perception is separated into two different parts throughout the Thesis: the perception phase and the decision-making phase. During the perception phase, the platform gathers information from the environment and builds an estimation of the actual state. A decentralized data fusion algorithm is proposed in the Thesis to combine data from heterogeneous entities and sensors. During the decision-making step, the teamed robots decide their courses of action in order to achieve certain long-term goals. These actions are performed taking into account the current state estimation, and they should entail some cooperation among the robots.

7.1.1 Decentralized data fusion

In Chapter 2, a probabilistic filter for decentralized data fusion is presented. Delayed states are considered in order to reach, in a decentralized manner, the same estimations that would be obtained by a central entity. In addition to this, the method presented enables the system to cope with communication delays and breakdowns. Under the assumption of Gaussian models and continuous state spaces, a beneficial structure of the information is also derived in order to reduce the communication bandwidth required to share estimations.

Apart from the above theoretical results, in Chapter 3 extensive field experiments are provided for decentralized data fusion with real robotic platforms. Thus, real implementations integrating heterogeneous sensors for cooperative tracking are presented. The consistency and robustness of the estimations as well as the improvements derived from the combination of different subsystems are demonstrated. Moreover, these robotic systems are tested under real communication constraints, such as wireless devices, bounded bandwidth, and data losses due to switches between access points.

Despite these theoretical and experimental results, the use of non-linear models can still cause some differences between decentralized and centralized estimations. Moreover, arbitrary network topologies may lead to inconsistent estimations, which can be prevented by using conservative fusion rules. Besides, the main drawback of the approach is that the whole development is based on the assumption of Gaussian distributions, which cannot consider multiple hypothesis. Other multi-modal distributions, such as Gaussian mixtures (Caballero et al., 2010), should be explored further in order to extend the algorithms proposed.

Furthermore, other decentralized data fusion schemes similar to the one proposed here were cited throughout the Thesis. However, the majority of them were found not to be able to either exactly reach the centralized solution for dynamic states or provide such successful results in field experiments using robotic platforms and communication constraints.

7.1.2 Cooperative decision-making

In Chapter 4, POMDPs are introduced as an option for decision-making approaches. They provide a well-founded mathematical framework that can deal with uncertainties, extending the data fusion methods mentioned above straightforwardly. Besides, they pursue long-term goals that can be adapted to a wide range of problems. In particular, this Chapter explains how POMDPs can be used for active perception, and POMDP-based models for multiple robots are presented. Unfortunately, POMDPs have also disadvantages, and some of them are tackled in the Thesis. For instance, most POMDP-based techniques in the literature for multiple robots require offline computation and only work with discrete and relatively small state spaces. In addition, existing models cannot be scaled in line with the number of robots at all, easily becoming unusable.

There are two main sources of non-scalability that arise naturally in POMDPs. First, the high-dimensional state spaces derived from large domains; and second, the exponential increase in the complexity of the model as more robots are added. Therefore, state-of-the-art optimal solutions for multi-robot POMDPs are feasible

only for small state spaces and require strong communication assumptions. Instead, this Thesis proposes approaches for multi-robot POMDPs that trade off optimality for applicability. Despite being sub-optimal, the achieved solutions can be used in real domains under unreliable communications, which are commonly found in robotic applications.

In particular, Chapters 5 and 6 tackle the source of intractability due to high-dimensional spaces. The assumption of considering part of the state observable reduces the complexity of the models by incorporating the notion of mixed observability. This idea is exploited in the Thesis in order to search for solutions that suit real robotic systems, where mixed observability can be reasonable for a wide range of applications involving active perception.

Chapter 5 proposes a closed solution for decentralized active perception with coordinated robots. Only single-robot POMDPs are solved for each individual robot without considering the others. This copes with the second source of complexity in multi-robot POMDPs and makes the approach scalable in line with the number of robots. Even though each robot pursues its own objectives, sharing information by means of a data fusion algorithm enables them to coordinate indirectly. For instance, in a tracking application, a robot unable to detect the target (e.g., because it is located far from it) could estimate the target location thanks to the information provided by the decentralized estimation.

However, a coordination technique in which each robot decides without taking into account its collaborators' actions still shows some drawbacks. In the example of the tracking application, if a robot knew what other members are doing, they would avoid visiting similar areas at the same time. This kind of explicit cooperation is attempted by the approach detailed in Chapter 6, where the problem is decomposed into parallel behaviors that can be auctioned in a decentralized manner. Thus, single-robot POMDPs are still solved (maintaining the scalability of the method), but there exist a direct cooperation to reach the common goal due to the distribution of behaviors.

Since the decentralized auction depends on the current belief and there is no a explicit consensus algorithm about it, different robots could have access to different estimations at the same time (due to latencies or errors in the data fusion approach)

and obtain inconsistent solutions. This concern and others related to data loss during the auction are studied in Chapter 6, which tests the robustness of the implementation in simulated and real robotic platforms. In any case, should they be necessary for certain applications, consensus algorithms could be incorporated to the system at a low communication cost.

Finally, despite the fact that some of the advantages for continuous Gaussian spaces derived in Chapter 2 cannot be exploited, the approaches presented in Chapters 5 and 6 use discrete state spaces because uni-modal distributions seriously degrade the performance of POMDPs. As mentioned before, other multi-modal distributions for continuous spaces may be studied to extend the methods of this Thesis.

7.2 Future developments

Future works should tackle the main drawbacks discussed in the previous Section. In regard to decentralized data fusion, the focus could be on further exploiting the information provided by the delayed states. For instance, in tracking applications, keeping a record of the trajectory of the target would provide more useful information when a prediction of its movement has to be made before planning actions. Techniques exploiting trajectory information could also be greatly helpful in order to cope with the track-to-track association problem. Moreover, extending that work to the multi-target case, new algorithms could be developed in order to deal with wrong associations made in the past by using the state trajectories.

Another issue is the fact that the centralized solution is no longer reached exactly when the information is fused in a conservative manner (e.g., the open field experiments proposed in Chapter 3). To address it, middlewares and communication protocols could be studied so that a tree-like network topology is enforced online for the system as new robots appear.

As mentioned before, future developments could consider other probabilistic distributions in the continuous space, such as particle filters or Gaussian mixtures, that can deal with multiple hypothesis.

In relation to the POMDPs, there are also several lines of future work. Planning under the assumption of continuous state spaces needs to be further explored. Even though there are some previous solvers of this kind, they do not offer the same level of performance and applicability as the discrete-based solutions. Moreover, the models involved in POMDPs are usually considered fixed, which may cause troubles in highly dynamic environments. Online approaches combined with active learning methods could be helpful to be able to recalculate these models as the system acts. In the same line, techniques to design the reward functions automatically would also be of interest.

Even though the lack of explicit cooperation in Chapter 5 is solved by the approach in Chapter 6, further development would still be desirable. In the method proposed by Chapter 6, no systematic technique exists to decompose the goals of the team into different behaviors. This distribution depends on each application, and the inclusion of a general method to do it would definitely broaden the applicability of the approach.

In addition to the ideas presented in this Thesis, there are other multi-robot approaches to control fleets of robots that could be investigated. For instance, better plans could be obtained if the actions of several robots were considered during the planning phase. A suggestion would be to include other robots' actions in the planning phase but trying to maintain vicinity (that is, to consider only the actions of potential neighboring robots instead of the whole fleet). This should lead to better policies while maintaining the scalability of the system.

Another proposal is to study multi-resolution POMDPs, i.e. POMDPs that work at different levels of abstraction in order to keep scalability. For instance, the decentralized auction proposed in this Thesis could be replaced by high-level POMDPs responsible for distributing the policies. Thus, low-level POMDPs would work with larger domains but a single robot, whereas high-level POMDPs would reason about several robots but in a reduced state space.

Appendix A

Resumen

La Tesis se centra en el estudio de técnicas de percepción cooperativa para equipos de múltiples robots. En nuestros días, existen muchas aplicaciones tanto exteriores como interiores en las que equipos formados por distintos robots pueden ser de gran utilidad. En estos casos, la complejidad de las tareas a realizar o la variabilidad de los lugares a los que hay que acceder, hacen que la cooperación entre robots heterogéneos sea imprescindible. En concreto, la Tesis trata situaciones de cooperación activa, en las que los robots no sólo se limitan a compartir información para estimar el estado del entorno, sino que toman decisiones conjuntas para mejorar dicha percepción.

La cooperación de múltiples sistemas también permite obtener un mayor grado de robustez, lo cual es relevante para operar en escenarios reales. En estos escenarios, existen importantes restricciones en las comunicaciones, por lo que la Tesis propone diferentes técnicas para lidiar con dichos problemas. Además, todas las soluciones propuestas son descentralizadas, ya que éstas son más fiables y escalables a medida que se aumenta el número de robots.

Adicionalmente, esta Tesis considera representaciones de estado y modelos probabilísticos. Una vez planteado el problema de percepción activa con múltiples robots de manera genérica, debido a su complejidad, se investigan soluciones aproximadas. De este modo, se intenta llegar a un compromiso entre optimalidad y aplicabilidad, ya que la Tesis está enfocada al funcionamiento en plataformas robóticas con fuertes restricciones de tiempo real.

Es importante mencionar que la Tesis presenta abundantes resultados experimentales con equipos de robots y sensores heterogéneos. Se incluyen tanto experimentos de campo como interiores que validan los métodos desarrollados en la Tesis para una aplicación de seguimiento cooperativo.

Los principales objetivos de la Tesis son los siguientes:

- Desarrollar un algoritmo descentralizado de fusion de datos con una representación común del estado para todos los robots. Los robots deben intercambiar información entre ellos y considerar modelos con incertidumbre.

- Estudiar métodos que solucionen los problemas que surgen al fusionar datos de manera descentralizada debidos a las comunicaciones. La convergencia de los métodos y las diferencias respecto a una versión centralizada se analizarán también para diferentes topologías de red.

- Cerrar el bucle desarrollando algoritmos de toma de decisiones cooperativas por parte de un equipo de robots. Estos algoritmos estarán basados en la capa de percepción cooperativa mencionada anteriormente.

- Analizar otros enfoques existentes en la literatura para cooperación descentralizada con incertidumbres. El objetivo es que los robots tengan que compartir la menor cantidad de información posible para poder cooperar.

- Las soluciones propuestas deben cumplir con los requisitos de escalabilidad del sistema necesarios para ser utilizado en plataformas robóticas reales.

- Validar los algoritmos en plataformas reales que realicen misiones cooperativas. Las implementaciones de los métodos desarrollados deben considerar aspectos de tiempo real y fiabilidad.

La Tesis está distribuida en varios capítulos cuyos contenidos se resumen a continuación:

- El Capítulo 1 introduce los objetivos de la Tesis y describe brevemente los proyectos de investigación a los cuales ha estado ligada directamente. También incluye un amplio estudio de trabajos similares pertenecientes al estado del arte.

- El Capítulo 2 presenta un método probabilístico para fusión descentralizada de datos. Se usan estados pasados de la estimación para converger hacia los mismos resultados que se obtendrían con una versión centralizada. A su vez, se desarrollan técnicas para reducir los efectos negativos de los fallos en las comunicaciones.

- El Capítulo 3 incluye experimentos del método desarrollado en el capítulo anterior en dos aplicaciones reales, robótica urbana y robótica para rescate. Los algoritmos se intengran en plataformas con robots reales que realizan seguimiento cooperativo de objetivos móviles. En concreto, se muestran la robustez y fiabilidad del sistema.

- El Capítulo 4 introduce los fundamentos matemáticos y el estado del arte de las técnicas POMDP, ya que en la Tesis se utilizan de forma asidua.

- El Capítulo 5 propone una aproximación coordinada para el problema de control de equipos de robots de forma descentralizada. Se combina la fusión descentralizada de datos con las técnicas POMDP para obtener comportamientos coordinados de manera implícita.

- El Capítulo 6 presenta una solución cooperativa para controlar múltiples robots. En este caso, además de la fusión descentralizada de datos, se realiza una distribución de tareas entre los robots que les permite cooperar de manera explícita.

- El Capítulo 7 resume las conclusiones de la Tesis y lanza ciertas líneas de trabajo futuro.

Bibliography

Aberdeen, D. and Baxter, J. (2002). Scaling internal-state policy-gradient methods for POMDPs. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 3–10.

AWARE (2006). Aware project web site. http://www.aware-project.net.

Bailey, T. and Durrant-Whyte, H. (2007). Decentralised data fusion with delayed states for consistent inference in mobile ad-hoc networks. Technical report, Australian Centre for Field Robotics, University of Sydney.

Balch, T. (2000). Hierarchic social entropy: An information theoric measure of robot group diversity. *Autonomous Robots*, 8(3):209–238.

Bar-Shalom, Y., Li, X., and Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation*. Wiley Interscience.

Becker, R., Zilberstein, S., Lesser, V., and Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840.

Bernstein, D. S., Hansen, E. A., and Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1282–1292.

Bertsekas, D. (2005). *Dynamic Programming and Optimal Control*. Athena Scientific.

Blanco, J., Gonzalez, J., and Fernandez-Madrigal, J. (2007). A consensus-based approach for estimating the observation likelihood of accurate range sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4032–4037.

Bonet, B. (2002). An epsilon-optimal grid-bsed algorithm for partially observable Markov decision processes. In *Proceedings of the 2002 International Conference in Machine Learning*, pages 51–58.

Bourgault, F. and Durrant-Whyte, H. (2004). Communication in general decentralized filters and the coordinated search strategy. In *Proceedings of The 7th International Conference on Information Fusion*, pages 723–730.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. (2004). Decentralized bayesian negotiation for cooperative search. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 2681–2686.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface.

Brooks, A., Makarenko, A., Williams, S., and Durrant-Whyte, H. (2006). Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems*, 54:887–897.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.

Burkard, R. E. (2002). Selected topics on assignment problems. *Discrete Applied Mathematics*, 123(1-3):257–302.

Caballero, F., Merino, L., Gil, P., Maza, I., and Ollero, A. (2008). A Probabilistic Framework for Entire WSN Localization Using a Mobile Robot. *Journal of Robotics and Autonomous Systems*, 56(10):798–806.

Caballero, F., Merino, L., and Ollero, A. (2010). A general gaussian-mixture approach for range-only mapping using multiple hypotheses. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, pages 4404–4409, Anchorage, Alaska (USA).

Capitan, J., Mantecon, D., Soriano, P., and Ollero, A. (2007). Autonomous perception techniques for urban and industrial fire scenarios. In *Proceedings of IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, Rome, Italy.

Capitan, J., Merino, L., Caballero, F., and Ollero, A. (2009). Delayed-State Information Filter for Cooperative Decentralized Tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3865–3870, Kobe, Japan.

Capitan, J., Merino, L., Caballero, F., and Ollero, A. (2011a). DDSIF: a new approach for cooperative decentralized tracking. *Robotics and Autonomous Systems*, In Press, Accepted Manuscript:–.

Capitan, J., Merino, L., and Ollero, A. (2010). Tracking under Uncertainty with Cooperating Objects using MOMDPs. In *The First International Workshop on Networks of Cooperating Objects (CONET)*, Stockholm, Sweden.

Capitan, J., Merino, L., and Ollero, A. (2011b). Multi-robot coordinated decision-making under mixed observability through decentralized data fusion. In *The 11th International Conference on Mobile Robots and Competitions.*

Capitan, J., Spaan, M., Merino, L., and Ollero, A. (2011c). Decentralized Multi-Robot Cooperation with Auctioned POMDPs. In *The 6th Workshop in Multiagent Sequential Decision Making in Uncertain Domains (MSDM). The 10th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS).*

Capitan, J., Spaan, M., Merino, L., and Ollero, A. (2011d). Decentralized Multi-Robot Cooperation with Auctioned POMDPs. In *Proceedings of the Robotics: Science and Systems Conference.* Submitted.

Cassandra, A. R., Littman, M. L., and Zhang, N. L. (1997). Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61.

Chang, H. S., Givan, R., and Chong, E. K. P. (2004). Parallel rollout for online solution of partially observable Markov decision processes. *Discrete Event Dynamic Systems*, 14(3):309–341.

Choi, H.-L., Brunet, L., and How, J. (2009). Consensus-Based Decentralised Auctions for Robust Task Allocation. *IEEE Transactions on Robotics*, 25:912–926.

Darrell, T. and Pentland, A. (1996). Active gesture recognition using partially observable Markov decision processes. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 984–988, Vienna, Austria.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

Dellaert, F., Burgard, W., Fox, D., and Thrun, S. (1999). Monte-Carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1322–1328, Detroit, USA.

Dempster, A. P. (1968). A Generalization of Bayesian Inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(2):205–247.

Derenick, J., Spletzer, J., and Hsieh, M. (2007). A graph theoretic approach to optimal target tracking for mobile robot teams. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 3422–3428.

Dias, M., Zlot, R., Kalra, N., and Stentz, A. (2006). Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270.

Duff, M. (2002). *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massassachusetts.

Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2004). Approximate Solutions For Partially Observable Stochastic Games with Common Payoffs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 136–143.

Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2005). Game theoretic control for robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1163–1169.

Eustice, R., Singh, H., and Leonard, J. (2006). Exactly Sparse Delayed-State Filters for View-Based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114.

Faugeras, O. and Luong, Q.-T. (2001). *The Geometry of Multiple Images: The laws that govern the formation of multiple images of a scene and some of their applications*. The MIT Press.

Fine, S., Singer, Y., and Tishby, N. (1998). The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1):41–62.

Foka, A. and Trahanias, P. (2007). Real-time hierarchical POMDPs for autonomous robot navigation. *Journal of Robotics and Autonomous Systems*, 55:561–571.

Forssén, P.-E. (2004). *Low and Medium Level Vision using Channel Representations*. PhD thesis, Linköping University. Thesis No. 858.

Furukawa, T., Dissanayake, G., and Durrant-Whyte, H. (2003). Time-optimal cooperative control for multiple robot vehicles. In *IEEE International Conference on Robotics and Automation*, pages 944–950.

Gerkey, B., Vaughan, R., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics, ICAR*, pages 317–323.

Gilbert, A., Illingworth, J., Bowden, R., Capitan, J., and Merino, L. (2009). Accurate fusion of robot, camera and wireless sensors for surveillance applications.

In *The Ninth IEEE International Workshop on Visual Surveillance, International Conference on Computer Vision, ICCV 2009*.

Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79.

Grime, S. and Durrant-Whyte, H. F. (1994). Data fusion in decentralized sensor networks. *Control Engineering Practice*, 2(5):849–863.

Grocholsky, B. (2002). *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, University of Sydney.

Grocholsky, B., Makarenko, A., Kaupp, T., and Durrant-Whyte, H. (2003). Scalable control of decentralised sensor platforms. In Zhao, F. and Guibas, L., editors, *Information Processing in Sensor Networks*, volume 2634 of *Lecture Notes in Computer Science*, pages 551–551. Springer Berlin / Heidelberg.

Guo, A. (2003). Decision-theoretic active sensing for autonomous agents. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics and Autonomous Systems*, pages 1002–1003.

Hansen, E. A. (1997). An improved policy iteration algorithm for partially observable MDPs. In *Proceedings of the Tenth Conference on Advances in Neural Information Processing Systems*, pages 1015–1021.

Hansen, E. A. (1998). Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 211–219.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Hauskrecht, M. (2000). Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–95.

Hauskrecht, M. and Fraser, H. (1998). Planning medical therapy using partially observable Markov decision processes. In *Proceedings of the International Workshop on Principles of Diagnosis*, pages 182–189.

Hoey, J. and Poupart, P. (2005). Solving POMDPs with continuous or large discrete observation spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1332–1338.

Hoey, J., Poupart, P., Bertoldi, A., Craig, T., Boutilier, C., and Mihailidis, A. (2010). Automated Handwashing Assistance for Persons with Dementia Using Video and a Partially Observable Markov Decision Process. *Computer Vision and Image Understanding*, 114(5):503–519. Special issue on Intelligent Vision Systems.

Hoey, J., St-Aubin, R., Hu, A., and Boutilier, C. (1999). SPUDD: Stochastic Planning Using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288.

Hsiao, K., Lozano-Perez, T., and Kaelbling, L. P. (2009). Relatively Robust Grasping. In *Workshop on Bridging the Gap Between Task and Motion Planning (ICAPS'09)*.

Hsieh, M. A., Cowley, A., Keller, J. F., Chaimowicz, L., andCamillo J. Taylor, B. G. V. K., End, Y., Arkin, R. C., Jung, B., Wolf, D. F., Sukhatme, G. S., and MacKenzie, D. C. (2007). Adaptive teams of autonomous aerial and ground robots for situational awareness. *Journal of Field Robotics*, 24:991–1014.

Ji, S. and Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5):1474–1485.

Ji, S., Parr, R., Li, H., Liao, X., and Carin, L. (2007). Point-based policy iteration. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, pages 1243–1249.

Julier, S. and Uhlmann, J. (1997). A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, volume 4, pages 2369–2373.

Jung, B. and Sukhatme, G. (2006). Cooperative Multi-robot Target Tracking. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 81–90. Springer.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

KaewTrakulPong, P. and Bowden, R. (2003). A Real-time Adaptive Visual Surveillance System for Tracking Low Resolution Colour Targets in Dynamically Changing Scenes. *Journal of Image and Vision Computing*, 21(10):913–929.

Kurniawati, H., Hsu, D., and Lee, W. (2008). SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of the Robotics: Science and Systems Conference*, Zurich, Switzerland.

LeBlanc, K. and Saffiotti, A. (2009). Multirobot Object Localization: A Fuzzy Fusion Approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(5):1259–1276.

Littman, M. L. (1996). *Algorithms for sequential decision making*. PhD thesis, Brown University.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the International Conference on Machine Learning*, pages 362–370.

Lovejoy, W. S. (1991). Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175.

Makarenko, A., Brooks, A., Kaupp, T., and Durrant-Whyte, H. (2009). Decentralised Data Fusion: A Graphical Model Approach. In *Proceedings of the 12th International Conference on Information Fusion*, pages 545–554.

Makarenko, A., Brooks, A., Williams, S., Durrant-Whyte, H., and Grocholsky, B. (2004). A decentralized architecture for active sensor networks. In *Proceedings of*

the IEEE International Conference on Robotics and Automation, ICRA, volume 2, pages 1097–1102.

Marron, P. J., Karnouskos, S., Minder, D., and Ollero, A. (2011). *The Emerging Domain of Cooperating Objects*. Springer.

Mathews, G., Durrant-Whyte, H., and Prokopenko, M. (2006). Scalable Decentralised Decision Making and Optimisation in Heterogeneous Teams. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2006)*, pages 383–388.

Mathews, G., Durrant-Whyte, H., and Prokopenko, M. (2008). *Advances in Applied Self-Organizing Systems*, chapter Decentralized Decision Making for Multiagent Systems. Springer-Verlag.

Maza, I., Caballero, F., Capitan, J., de Dios, J. M., and Ollero, A. (2010). Firemen Monitoring with Multiple UAVs for Search and Rescue Missions. In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*.

Maza, I., Caballero, F., Capitan, J., de Dios, J. M., and Ollero, A. (2011a). A Distributed Architecture for a Robotic Platform with Aerial Sensor Transportation and Self-Deployment Capabilities. *Journal of Field Robotics*. To appear.

Maza, I., Caballero, F., Capitan, J., de Dios, J. M., and Ollero, A. (2011b). Experimental Results in Multi-UAV Coordination for Disaster Management and Civil Security Applications. *Journal of Intelligent and Robotic Systems*, 61:563–585.

Merino, L. (2007). *A cooperative perception system for multiple unmanned aerial vehicles. Application to the cooperative detection, localization and monitoring of forest fires*. PhD thesis, University of Seville.

Merino, L., and. J.R. Martínez-de Dios, F. C., Ferruz, J., and Ollero, A. (2006). A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires. *Journal of Field Robotics*, 23:165–184.

Meuleau, N., Kim, K., Kaelbling, L. P., and Cassandra, A. R. (1999a). Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 417–426.

Meuleau, N., Peshkin, L., Kim, K., and Kaelbling, L. P. (1999b). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 427–436.

Monahan, G. E. (1982). A survey of partially observable Markov decision processes: theory, models and algorithms. *Management Science*, 28(1):1–16.

Mosteo, A. and Montano, L. (2007). Comparative Experiments on Optimization Criteria and Algorithms for Auction based Multi-Robot Task Allocation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3345–3350.

Nair, R., Tambe, M., Roth, M., and Yokoo, M. (2004). Communication for improving policy computation in distributed POMDPs. In *Proceedings of International Conference on Autonomous Agents and Multi Agent Systems*, pages 1098–1105.

Nair, R., Varakantham, P., Tambe, M., and Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the National Conference on Artificial Intelligence*, pages 133–139.

Nash, John F., J. (1950). The bargaining problem. *Econometrica*, 18(2):pp. 155–162.

Nettleton, E. (2003). *Decentralised Arquitectures for Tracking and Navigation with Multiple Flight Vehicles*. PhD thesis, University of Sydney.

Nettleton, E., Durrant-Whyte, H., and Sukkarieh, S. (2003). A Robust Architecture for Decentralised Data Fusion. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*.

Ng, A. and Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415.

Oliehoek, F. A., Spaan, M. T., and Vlassis, N. (2008). Optimal and Approximate Q-value Functions for Decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353.

Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. (2009). POMDPs for Robotic Tasks with Mixed Observability. In *Proceedings of the Robotics: Science and Systems Conference*, Seattle, USA.

Pahliani, A. and Lima, P. (2007). Cooperative Opinion Pool: A New Method for Sensor Fusion by a Robot team. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 3721–3726, San Diego, USA.

Paquet, S., Chaib-draa, B., and Ross, S. (2006). Hybrid POMDP algorithms. In *Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, pages 133–147.

Paquet, S., Tobin, L., and Chaib-draa, B. (2005). An online POMDP algorithm for complex multiagent environments. In *Proceedings of The fourth International Joint Conference on Autonomous Agents and Muiti Agent Systems*, pages 970–977.

Parker, L. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, (12):231–255.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1025–1032.

Pineau, J., Gordon, G., and Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380.

Pineau, J., Roy, N., and Thrun, S. (2001). A hierarchical approach to POMDP planning and execution. In *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML 2001)*.

Pirjanian, P. and Mataric, M. (2000). Multi-robot target acquisition using multiple objective behavior coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2696–2702.

Porta, J. M., Vlassis, N., Spaan, M. T., and Poupart, P. (2006). Point-based value iteration for continuous POMDPs. *The Journal of Machine Learning Research*, 7:2329–2367.

Poupart, P. (2005). *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto.

Prentice, S. and Roy, N. (2009). The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465.

Pynadath, D. V. and Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423.

Rosencrantz, M., Gordon, G., and Thrun, S. (2003). Decentralized sensor fusion with distributed particle filters. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, pages 493–500.

Ross, S. and Chaib-draa, B. (2007). AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 2592–2598.

Ross, S., Pineau, J., and S., P. (2008). Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704.

Roth, M., Simmons, R., and Veloso, M. (2005). Decentralized communication strategies for coordinated multi-agent policies. In Schultz, A., Parker, L., and Schneider, F., editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume IV. Kluwer Academic Publishers.

Roy, N., Gordon, G., and Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40.

Sanfeliu, A. and Andrade-Cetto, J. (2006). Ubiquitous networking robotics in urban settings. In *Workshop on Network Robot Systems. Proceedings of the 2006 IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Sanfeliu, A., Andrade-Cetto, J., Barbosa, M., Bowden, R., Capitan, J., Corominas, A., Gilbert, A., Illingworth, J., Merino, L., Mirats, J., Moreno, P., Ollero, A., Sequeira, J., and Spaan, M. (2010). Decentralized Sensor Fusion for Ubiquitous Networking Robotics in Urban Areas. *Sensors*, 10:2274–2314.

Seuken, S. and Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17:190–250.

Shafer, G. (1976). *A mathematical theory of evidence*. Princetion University Press, Princetion, NJ.

Shani, G., Brafman, R., and Shimony, S. (2005). Adaptation for changing stochastic environments through online POMDP policy learning. In *Proceedings of the Workshop on Reinforcement Learning in Non-Stationary Environments*, pages 61–70.

Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

Smith, T. and Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 520–527.

Smith, T. and Simmons, R. (2005). Point-based POMDP algorithms: improved analysis and implementation. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*, pages 542–547.

Sondik, E. J. (1971). *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University.

Sondik, E. J. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26:282–304.

Spaan, M., Gonçalves, N., and Sequeira, J. (2010). Multirobot coordination by auctioning pomdps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1446–1451.

Spaan, M. T. J. (2008). Cooperative active perception using POMDPs. In *AAAI 2008 Workshop on Advancements in POMDP Solvers*.

Spaan, M. T. J., Oliehoek, F. A., and Vlassis, N. (2008). Multiagent planning under uncertainty with stochastic communication delays. In *International Conference on Automated Planning and Scheduling*, pages 338–345.

Spaan, M. T. J. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

Spletzer, J. and Taylor, C. (2003). Dynamic sensor planning and control for optimally tracking targets. *International Journal of Robotics Research*, 22:7–20.

St-Aubin, R., Hoey, J., and Boutilier, C. (2001). APRICODD: Approximate Policy Construction using Decision Diagrams. In *Advances in Neural Information Processing Systems*, pages 1089–1095.

Stone, L. D., Corwin, T. L., and Barlow, C. A. (1999). *Bayesian Multiple Target Tracking*. Artech House, Inc., Norwood, MA, USA.

Stroupe, A. and Balch, T. (2005). Value-based action selection for observation with robot teams using probabilistic techniques. *Robotics and Autonomous Systems*, 50:85–97.

Sukkarieh, S., Nettleton, E., Kim, J.-H., Ridley, M., Goktogan, A., and Durrant-Whyte, H. (2003). The ANSER Project: Data Fusion Across Multiple Uninhabited Air Vehicles. *The International Journal of Robotics Research*, 22(7-8):505–539.

Szer, D., Charpillet, F., and Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 576–583.

Theocharous, G. and Mahadevan, S. (2002). Approximate planning with hierarchical partially observable Markov decision processes for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1347–1352.

Theocharous, G., Murphy, K., and Kaelbling, L. P. (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1045–1051.

Theocharous, G., Rohanimanesh, K., and Mahadevan, S. (2001). Learning Hierarchical Partially Observed Markov Decision Process Models for Robot Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 511–516.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *Internation Journal of Robotics Research*, 23(7–8):693–716.

Toussaint, M., Charlin, L., and Poupart, P. (2008). Hierarchical POMDP Controller Optimization by Likelihood Maximization. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 55–60.

Uhlmann, J. K. (2003). Covariance consistency methods for fault-tolerant distributed data fusion. *Information Fusion*, (4):201–215.

Ulrich, I. and Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1572–1577.

Utete, S. and Durrant-Whyte, H. (1994). Reliability in decentralised data fusion networks. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 215–221.

Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., and Yokoo, M. (2007). Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proceedings of International Conference on Autonomous Agents and Multi Agent Systems*, pages 218:1–218:8.

Viguria, A., Maza, I., and Ollero, A. (2008). S+T: An Algorithm for Distributed Multirobot Task Allocation based on Services for Improving Robot Cooperation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3163–3168.

Villamizar, M., Scandaliaris, J., Sanfeliu, A., and Andrade-Cetto, J. (2009). Combining color invariant gradient detector with HOG descriptors for robust image detection in scenes under cast shadows. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1997–2002.

Viola, P. and Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57:137–154.

Vlassis, N., Gordon, G., and Pineau, J. (2006). Planning under uncertainty in robotics. *Robotics and Autonomous Systems*, 54(11). Special issue.

Waslander, S. L., Inalhan, G., and Tomlin, C. J. (2003). *Theory and Algorithms for Cooperative Systems*, chapter Decentralized Optimization via Nash Bargaining, pages 565–583.

Wong, E.-M., Bourgault, F., and Furukawa, T. (2005). Multi-vehicle Bayesian search for multiple lost targets. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3169–3174.

Yanli, Y., Minai, A., and Polycarpou, M. (2005). Evidential map-building approaches for multi-UAV cooperative search. In *Proceedings of the American Control Conference*, pages 116–121.

Zadeh, L. A. (1999). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100:9–34.

Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51.

Zhao, F., Shin, J., and Reich, J. (2002). Information-driven dynamic sensor collaboration. *Signal Processing Magazine*, 19(2):61–72.

Zhou, R. and Hansen, E. A. (2001). An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 707–716.

Zlot, R., Stentz, A., Dias, M., and Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3016–3023.