Cooperative POMDPs for human-Robot joint activities

Fabio Valerio Ferrari

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité *Informatique*

Préparée au sein de l'Université de Caen Normandie

Cooperative POMDPs for Human-Robot Joint Activities

Présentée et soutenue par
Fabio-Valerio FERRARI

Thèse soutenue publiquement le 14/12/2017 devant le jury composé de		
M. Rachid ALAMI	Directeur de Recherche, CNRS, Toulouse	Rapporteur
M. Raja CHATILA	Directeur de Recherche, CNRS, Université de Paris 6	Rapporteur
M. Abdel-Illah MOUADDIB	Professeur des Universités, Université de Caen Normandie	Directeur de thèse
M. Luca IOCCHI	Professeur des Universités, Università di Roma La Sapienza	Examinateur
M. Laurent JEANPIERRE	Maitre de Conférence, Université de Caen Normandie	Examinateur
M. Laurent VERCOUTER	Professeur des Universités, INSA de Rouen	Examinateur

Thèse dirigée par Abdel-Illah MOUADDIB, laboratoire GREYC

UNIVERSITÉ CAEN NORMANDIE

GREYC

Acknowledgements

Je tiens tout d'abord à remercier chaleureusement Abdel-Illah Mouaddib et Laurent Jeanpierre pour m'avoir encadré durant cette thèse. Je les remercie pour le temps et effort qu'ils ont consacré à corriger cette thèse et pour leur précieux enseignements. C'est notamment grace aux nombreuses heures passées dans le bureau de Laurent et à son immense patience que j'ai pu apprendre de mes erreurs et prendre confiance dans le domaine de l'IA.

Je remercie Rachid Alami, Raja Chatila, Luca Iocchi et Laurent Vercouter pour avoir accepté de faire partie du jury et d'avoir étudié mon travail.

Je remercie Quentin, Mathieu et Jonathan avec qui j'ai partagé une bonne partie de ces trois ans de thèse. Merci pour la bonne ambiance, les discussions, les pauses au RU et à la perm Optic, et les soirées jeu. Merci aussi pour tous les conseils et l'aide donnés. Je remercie aussi Flo, Jenny, Hichem, Simone, et tous les autres amis et amies thésards avec qui j'ai passé des bons et précieux moments.

Enfin, un grand merci à ma famille, toujours prete à parcourir des centaines de kilometres pour etre ensemble. Merci à mes parents, Anna et Luciano, pour le soutien porté dans les moments plus difficiles, et à mon frère, Victor, pour les nombreuses heures de jeu et discussions.

Walking with a friend in the dark is better than walking alone in the light.

— Helen Keller

Résumé

Objectif de la thèse est le développent de méthodes de planification pour la résolution de tâches jointes homme-robot dans des espaces publiques. La robotique de service dans les espaces publiques comme les musées, les aéroports et les centres commerciaux est un domaine en forte croissance. Cependant, les utilisateurs qui coopèrent avec le robot peuvent facilement se distraire et abandonner la tâche jointe. La thèse se focalise donc sur les défis posés par l'incertitude et imprévisibilité d'une coopération avec un humain. La thèse décrit l'état de l'art sur la coopération homme-robot dans la robotique de service, et sur les modèles de planification. Elle présente ensuite une nouvelle approche théorique, basée sur les processus décisionnels de Markov partiellement observables, qui permet de garantir la coopération de l'humain tout au long de la tâche, de façon flexible, robuste et rapide. La contribution majeure de la thèse consiste en une structure hiérarchique qui sépare l'aspect coopératif d'une activité jointe de la tâche en soi. L'approche a été appliquée dans un scénario réel, un robot guide dans un centre commercial. La thèse présente les expériences effectuées pour mesurer la qualité de l'approche proposée, ainsi que les expériences avec le robot réel.

Abstract

The number of applications of service robotics in public spaces such as hospitals, museums and malls is a growing trend. Joint activities between human and robot agents in public spaces, however, provide several challenges to the robot, and specifically with its planning capabilities: they need to cope with a dynamic and uncertain environment and are subject to particular human-robot interaction constraints. A major challenge is the Joint Intention problem. When cooperating with humans, a persistent commitment to achieve a shared goal cannot be always assumed, since they have an unpredictable behavior and may be distracted in environments as dynamic and uncertain as public spaces.

The thesis presents a novel method for ensuring cooperation between human and robot, even with the uncertainty of the human's behavior, and ensuring a robust execution of policies. The approach consists in the development of a hierarchical and flexible framework based on POMDPs. The framework partitions the overall joint activity into independent planning modules, each dealing with a specific aspect of the joint activity: either ensuring the human-robot cooperation, or proceeding with the task to achieve itself. The cooperation part can be solved independently from the task and executed as a finite state machine in order to contain online planning effort. In order to do so, however, we introduce a *belief shift* function and describe how to use it to transform a POMDP policy into an executable finite state machine.

The developed framework has been implemented in a real application scenario as part of the COACHES project. The thesis describes the Escort mission used as testbed application and the details of implementation on the real robots. This scenario has as well been used to carry several experiments and to evaluate our contributions.

Contents

List of Figures

List of Abbreviations

HRI	**H**uman **R**obot **I**nteraction
MDP	**M**arkov **D**ecision **P**rocess
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocess
BMDP	**B**elief **M**arkov **D**ecision **P**rocess
DBMDP	**D**iscrete **B**elief **M**arkov **D**ecision **P**rocess
MOMDP	**M**ixed **O**bservability **M**arkov **D**ecision **P**rocess
HMM	**H**idden **M**arkov **M**odel
DBN	**D**ynamic **B**ayesian **N**etwork
FSM	**F**inite **S**tate **M**achine
PNP	**P**etri **N**et **P**lan
JI	**J**oint **I**ntention

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

As robotic technologies continue to evolve, more robots are being used to provide assistance to humans in public spaces. Such kind of application, however, presents several challenges that the robot has to face. This thesis addresses the design of a decision-making framework for service robots cooperating with humans in public spaces. In particular, we believe that the robot needs to account for the unpredictability of human behavior and his low level of commitment when cooperating in a joint activity.

In many applications, when a robot cooperates with humans, the commitment of all agents to the shared task is assumed to be persistent. While this assumption may be appropriate when cooperating with professional workers, it may not hold with service applications in public environments, such as malls, museums, and airports. In these domains, the users that the robot has to cooperate with are untrained users, customers passing by and bystanders that can be distracted by the dynamic environment. It has been shown that users in public spaces prefer short-term interactions with robots rather than long-term ones [1], and that they may stop interacting with the robot if it doesn't draw their attention [2]. Instead of assuming the full commitment to the task of the human agent or treating the lack of engagement as an unexpected environmental factor, or even as a failure, the user's level of attention should be accounted for by the planning and decision-making capabilities of the robot. We call this issue the *Joint Intention problem*: robots cooperating with humans should be able not only to achieve the shared

goal, but also to ensure that the human-robot team is jointly committed to cooperate all along the task. This problem is an ongoing field of research that has been addressed only recently in Literature.

1.2 Running Example: Escort task

The concepts and contributions described in this thesis will be illustrated trough a single example scenario. This scenario is the testbed application of the European project COACHES[1]. It consists of the deployment of a couple of service robots, named Cadomus and Romus in a shopping mall in Caen.

1.2.1 The COACHES project

Public spaces in large cities are progressively becoming unwelcoming and difficult to use because of the overcrowding and complex information in signboards. It is in the interest of cities to make their public spaces easier to use, friendlier to visitors and safer to increasing elderly population and to citizens with disabilities. The development of robotic technologies, can provide in the near future teams of robots to be deployed in public spaces in order to accomplish services that can help humans.

To this end, the COACHES project addresses fundamental issues related to the design of a network of autonomous robots with high- level skills of environment modeling and scene understanding, distributed autonomous decision-making, short-term interacting with humans and robust and safe navigation in overcrowding spaces. The modular architecture developed within the project features a knowledge-based representation of the environment, human activities and needs estimation using Markov and Bayesian techniques, distributed decision-making under uncertainty to collectively plan activities of assistance, guidance and delivery tasks using Decentralized Partially Observable Markov Decision Processes, and a multi-modal and short-term human- robot interaction to exchange information and requests.

Several partnerships are involved in the COACHES project. The project is developed trough the collaboration of University of Caen (UCBN), University La Sapienza of Rome

[1] `https://coaches.greyc.fr/`

(Sapienza), the Sabienci University of Istanbul (SU) and Vrije University of Bruxelles (VUB). Each university provides complementary competences from cognitive systems (SU), robust image/video processing (VUB, UCBN), and semantic scene analysis and understanding (VUB), Collective decision-making using decentralized partially observable Markov Decision Processes and multi-agent planning (UCBN, Sapienza), multi-modal and short-term human-robot interaction (Sapienza, UCBN). The end-user "Caen la mer"[2] provides the scenarios where the COACHES robots and systems are deployed: a mall called "Les Rives de l'Orne" in the city of Caen (France).

The project deploys a set of static network sensor (cameras) perceiving the environment and providing the relevant features, and a set of robots perceiving their surroundings through their on-board sensor and in cooperation with the external sensor (cameras) and proposing assistance services to the neighbouring people. This system is expected to be deployed in a mall to assist visitors and shopkeepers, and to support the mall managers for surveillance and security. To do so, COACHES provides an integrated solution to new challenges on:

1. A rich representation and reasoning techniques for modeling a changing environment. This representation E describes objects and their spatial relations as a reference description of the space which will be compared to the perceived environment E' to detect abnormal objects or human activities. The difference between E and E' allows the robot to generate events of abnormal situation.

2. Sophisticated probabilistic reasoning perceiving the environment E' using external cameras with a global but imprecise view of the space and local sensor of the robots for a local and accurate view. We develop on-board real-time multi-sensors computer vision approaches for recognizing complex human activities and event analysis, as well as contextual relationships of objects in the scene.

3. A multi-modal and short-term human-robot interaction to exchange information and requests. COACHES robots communicate with users via touch, text and speech interpretation but also via static screen of the mall. The robots are able to answer to queries on destinations by providing the route on the map, by guiding a person to a destination, or transporting their bags to the requested destination. We consider three kinds of interaction: with visitors, shopkeepers and mall's managers.

[2] `http://www.caenlamer.fr/`

4. Distributed decision-making under uncertainty and learning to collectively plan activities of assistance, guidance and delivery tasks using Decentralized Partially Observable Markov Decision Processes with efficient algorithms with high scalability and adapt their assistance from their interactions and their navigation to the current situation and the overcrowding areas.

In addition to the scientific research on the above mentioned topics, we are also interested in the scientific results obtained by integrating the developed solutions and by evaluating the overall system in real world environments. To this end, the technical solutions that are investigated within these scientific objectives are developed and integrated in a modular architecture and validated through several use cases defined by an end-user in the "Rives de l'Orne" mall of the Caen city. These use cases are dedicated to demonstrate different tasks provided by the robots to assist the visitors. These demonstrations will include the following functionalities:

1. informing visitors by displaying advertisement, providing maps, etc.
2. guiding visitors by displaying a path in the screen, pointing to the destination moving towards the target destination
3. surveillance to acquire information about the environment as requested by the mall's managers, as well as automatic detection of abnormal situations in the environment.

The overall architecture of the COACHES system is shown in Figure 1.1. It consists of several modules developed by the different partners universities of the project. As the scope of the thesis lies within the planning and decision-making capabilities of the robot, the contributions presented in this work belong mostly to the *Multi-robot cooperative planning under uncertainty* module (WP4), with minor contributions in the *Situation Awareness* component (T22). The main modules of the architecture and how they relate to each other will be described more in detail in Chapter 5.

1.2.2 The Escort task

The robots have several tasks to perform in the shopping mall:

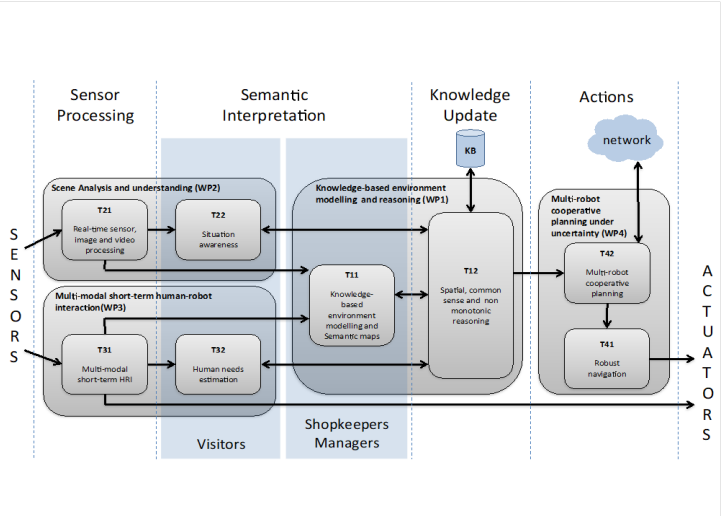


FIGURE 1.1: The COACHES software architecture

- *Advertise*: The robot wanders and shows advertisements about the mall and the shops. This is mostly an idle behavior when no other task is scheduled.
- *Assist*: The robot interacts with customers and provides any kind of help or information requested. The robot has a proactive behavior and may initiate this task when it detects new customers at the mall's entrance.
- *Escort*: This task consists of the robot guiding and escorting the user to a selected point of interest (POI). This is seen as a cooperative task between the human and the robot.
- *Patrol*: The robot patrols around a specific area for security reasons. This task may be requested by the mall staff, for instance when there is work in progress.

Among these, we will focus on the Escort task. We consider it as a joint task where both the human and the robot have to cooperate to achieve their common goal, that is, reach the desired destination. The robot does not only have to lead the user towards a goal position, but also to react to his behavior and to ensure that the commitment to the joint task is preserved. As a guided customer in the shopping mall, all along the escort task the person may look around, focus his attention to the shops and persons nearby or stop unexpectedly to do some urgent activity. This lack of attention may cause the human to loose track of the robot and get lost, or to change his mind and head towards a different destination. Therefore, the Escort task is a pertinent application domain to highlight the challenges of the Joint Intention problem.

Nevertheless, the Joint Intention problem may apply to any kind of human-robot collaboration in public spaces. Although we only use the Escort task as an illustrative scenario and as a testbed application for the real robots, the contributions of this thesis do not limit themselves to guide robots. The developed approach can be easily used with different cooperative tasks.

Also, this thesis will focus on single robot and single user tasks. Although the broader goals of the COACHES project include the development of decentralized decision-making frameworks for the robot team, multi-agent planning is beyond the scope of the thesis. This thesis only includes a brief discussion on the extension of our work to Group Escorts, where the robot guides a group of users.

1.3 Outline

The thesis is organized as follows. Part II presents a review of the Literature on human-robot interaction and on planning models and algorithms. Part III describes the main contributions of the thesis: both the theoretical framework for planning human-robot cooperation and its implementation in the example scenario. It also presents several experiments performed, both in simulation and on real robots, and discuss the results. Part IV provides the conclusions of the thesis.

Literature Review

- **Chapter 2**: in this Chapter we provide a non-exhaustive review on human-robot interaction, and specifically human-robot cooperation with service robots in public spaces. We describe the challenges of public spaces as application environments and we present methods and formalisms adopted in Literature to model the human's cooperation and exploit it. We also provide a review of guide robots to contextualize the Escort task scenario used in the thesis as application example.
- **Chapter 3**: this Chapter reviews the state of the art in planning models. We briefly review classic planning models, then we focus on Markov Decision Processes, which are a successful approach for probabilistic planning. We define the model and present the main resolution techniques and a few extensions. Similarly,

we define the Partially Observable Markov Decision Processes, and review their algorithms and extensions.

Contributions

- **Chapter 4**: in this Chapter we describe the theoretical framework developed for dealing with the human's cooperation when planning a human-robot joint activity. We describe the hierarchical structure of the framework and detail how to decompose the joint activity into two separate and independent sub-systems: the Task and Cooperation modules. We also provide a method for solving the POMDP that models the Cooperation sub-system: we introduce a *belief shift* function and use it to translate a discrete Belief-MDP policy into an executable POMDP policy.
- **Chapter 5**: this Chapter details the Escort task application scenario, and contextualizes the proposed framework within the COACHES project architecture. We describe in detail the implementation of the Cooperation POMDP and its execution process on the Cadomus and Romus robots.
- **Chapter 6**: we describe in this Chapter the experiments performed to evaluate our approach. We introduce the performance criteria and compare our resolution technique with state of the art algorithms. We describe the simulation environment and show the performance results obtained. Then we show the experiments performed with the real robot.

Conclusions

We conclude the thesis by summarizing the contributions provided and by discussing the perspectives of future development of this work.

Part II

Literature Review

Chapter 2

Human Robot Interaction

Human-Robot Interaction (HRI) is a field of study dedicated to understanding, designing and evaluating robotic systems for use by or with humans [3]. This field is a growing trend in robotics, and the past decade has seen a considerable increase of the number of studies and applications in HRI domains, and of the media's attention as well. Among the many fields of research of HRI, we will focus on service robots deployed in public spaces that interact and cooperate with human users. This chapter presents a review of service robot applications, specifically guide robots, and of models and methods developed for coping with the challenges of public environments and human-robot cooperation.

2.1 Overview of HRI

Human-Robot Interaction is a very wide spanning multi-disciplinary domain. HRI greatly benefits from research on vision and image processing, sensor fusion, artificial intelligence, learning, theory of mind, robot design and natural language processing, but also from psychology and social sciences. As a whole, HRI may include a great variety of applications very different from each other. A review on HRI applications and key features is provided in [4], [3] and [5]. In this thesis, we will focus only on a few key aspects of HRI, notably the use of robots in public environments and the cooperation with humans in a joint human-robot activity.

2.2 Service Robots in Public Spaces

Service robots are designed to support and service humans trough physical and social interactions. Their use, especially in professional environments, is a growing trend and the range of applications is increasing. It is in the interest of public and private administrations to make their public spaces easier to use, friendlier to visitors and safer to an increasing elderly population and to citizens with disabilities. Ivanov et al. [6] provide a review of current uses of robots in tourism and hospitality domains and highlight the economic interest of service robots. In the last two decades, service robots have been deployed, for instance, in museums and exhibitions [7][8][9][10][11][12][13], shopping malls [14][15][16][17], nursing homes [18] and airports [19].

In order to be incorporated into human populated environments, such as homes, workplaces and public service facilities, a robot needs not only to be safe, but, for greater success, to be "social" as well. Fong et al. [5] define three classes of robot sociability:

- *Socially situated*: socially situated robots perceive and react to other social agents differently from objects in the environment.
- *Socially embedded*: socially embedded robots are socially situated and at least partially aware of human interaction structures and social rules.
- *Socially intelligent*: socially intelligent robots possess deep models of human cognition and are able to mimic human social intelligence.

Much research has been performed, both from robotics and social sciences, to analyze the social rules and structures that naturally arise in human-human interactions, so that robots may understand them, implement them, and eventually reason about them.

2.2.1 Challenges of public spaces

Autonomous robots have to face several issues in order to perform their task in crowded public spaces.

FIGURE 2.1:
A shopping mall. Such kind of environments presents several challenges for service robots.

Dynamic Environment

The real-world is a very challenging domain for robots, especially unrestricted spaces such as malls, hospitals and airports. The physical environment is dynamic and unpredictable, and it may change at any time. It is populated by many individuals, sometimes even by crowds, whether they are workers, customers or bystanders. Objects such as pieces of furniture or equipment may be moved, removed or introduced unexpectedly. Additionally, input sensors have to cope with dynamic occlusions and a higher level of noise with respect to restricted and controlled environments.

Such challenging conditions require that the robot ensures an high degree of robustness for its video and audio processing capabilities, as well as mapping and localization. However, its decision-making capabilities require soundness too: the robot's plan needs to be able to account for the dynamic and unpredictable environment in a fast and robust way.

Socially-aware navigation

In order to be socially accepted and to operate more naturally in a human environment, a robot should take particular care when navigating and working in a physical space shared with humans. More specifically, it should adopt social conventions that emerge

naturally between humans. The first study on social spaces, known as *Proxemics*, was performed by Hall [20]. His work shows how, for each person, it is possible to identify four regions of space, centered around him, which are associated to the comfort and social acceptance of distance from other persons. The four regions, and the associated distances that define them, are the following:

- *Intimate*: between 0 and 46 cm.
- *Personal*: between 46 and 122 cm.
- *Social*: between 1.2 and 3.7 m.
- *Public*: between 3.7 and 7.6 m. (and beyond)

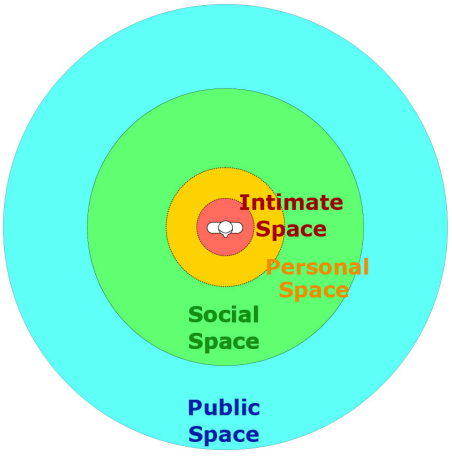


FIGURE 2.2: Social spaces in Proxemics

The described social spaces depend on parameters that may vary according to context, age, task, and personality [21]. The social regions may therefore vary in shape and size: for instance, Pandey and Alami [12] use ellipsoid regions instead of circular ones. Other fields of study include adapting the robot's speed to be more socially acceptable [22] and how to approach a person or a group of persons in a fluent, natural and comfortable way [12][23]. These studies have been applied in several robotic applications. A survey on human-aware navigation is given by Kruse et al. [24].

Detection and Tracking

To interact with a person, a robot first needs to detect him. Depending on the applications, this may be as simple as waiting for a button to be pressed. Personal assistant and cognitive assistant robots may recognize verbal input to start their interaction. Several other applications, however, especially those where human and robot operate in a shared physical space, require that the robot detects the human's position, posture and/or face. Not only this is a necessary step for several tasks, such as following a person or handing him an object, but it may provide additional benefits. First, by tracking the person's position, posture or gaze, it is possible to understand the person's activity or to estimate his state of mind. Second, if the person is not only detected, but identified as well, the robot is able to provide a personalized assistance [14]. Third, even in those applications where it is not required, improving the detection and tracking of the persons it is interacting with may increase the sociability and friendliness of the robot. Pitsch et al. [2], for instance, show how users react differently depending on the robot's head orientation.

Because of the uncertainty of the dynamic environment and the quantity of persons passing by in public spaces, detecting and tracking users is no trivial task.

Several methods have been adopted. Kanda et al. [14] provide users with RFID (Radio-Frequency IDentification) tags to identify and track them; Arras et al. [25][26] use laser scans to recognize people's feet; Zhang et al. [27] uses visual features to detect people in 2D camera videos, while Jafari et al. [28] use both visual and depth information to track persons.

Dialogue

Communication is an essential part of human-robot interaction, especially in joint activities. A robot may communicate trough spoken dialogue, gestures, display, and even facial expressions. Even simple head and gaze orientations have been used to convey information [29]. For spoken dialogue, communication models and social rules such as turn-taking, have been investigated [30]. Trough dialogue, the robot may attempt to understand what the human is doing and what he wishes to do, so that it may better help him. An excessive use of dialogue, however, may be considered annoying and unnatural for the human. To be more socially acceptable, a robot should be able to infer

the state of the human trough passive observation and reduce the use of queries (as in, for example, [31]). The inference of the human's intentions and level of attention will be discussed in detail in Section 2.3.

2.2.2 Robots Guides

In this section we review on the main application of this thesis and one of the most common tasks of service robots in public spaces: to provide information and guide users towards desired destinations. A highly popular application domain in HRI is a robot guide in a museum, probably because the environmental and interaction conditions of museums are relatively stable and controlled. Rhino[7] and Minerva[1] have been the pioneers of a long series of tour-guide robots. Rhino operated for six days at the Deutsches Museum Bonn in Germany, guiding real visitors and virtual visitors trough a Web interface. Rhino required that the map of the museum had been provided before-hand. Minerva, instead, could perform online mapping of the environment, specifically the Smithsonian's National Museum of American History in the United States where it was deployed. Minerva's operation in the museum showed that in public spaces people tend to perform short-term interactions with the robot, instead of long-term interactions more commonly found in other application fields. Another museum guide robot, called Chips [13], provided interesting insight during its deployment at the Carnegie Museum of Natural History in Pittsburgh, Pennsylvania (U.S.). The authors remarked how the persons' interest and attention increased when the robot was in motion and when it showed a proactive behavior. In particular, users showed surprise and interest when the robot autonomously initiated dialogue, and when it exhibited limited prosody during long, static presentations in the museum. While the general public may gradually become more accustomed to proactive robots with respect to the previous decade, it still stands to reason that a robot in motion will draw more attention than a static object.

Many works focused on the interactive aspects of the tour-guide robot. RoboX [32], for instance, features speech recognition, face detection and tracking of users, and an emotional state machine capable of expressing different emotions trough a LED matrix. Urbano [8] presents similar features, but uses hand gestures and a robotic face to express emotions. Even more so, Kanda et al [14] forego autonomous navigation to enhance the interaction experience. Their humanoid robot Robovie stands in a fixed location of a

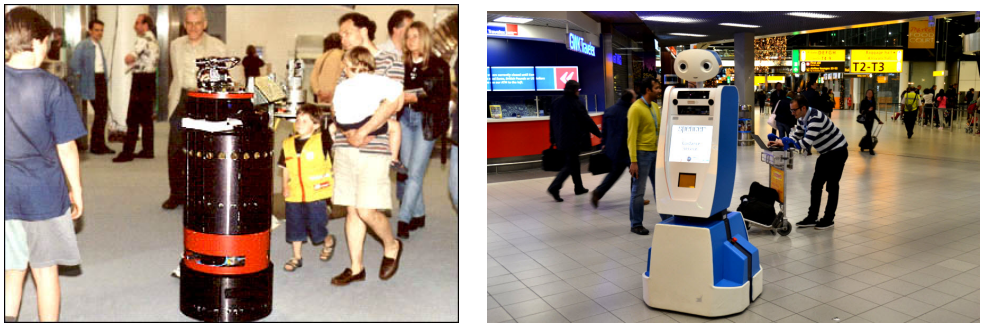
shopping mall, and uses floor sensors and RFID tags provided to the participants to detect and identify them. This allows it to provide more personalized, friendly and natural interactions, but constraints it to guide the customers using only gestures and verbal directions.

Other works, on the opposite, focus on implementing an efficient navigation and guidance for humans. Martinez et al. [33], for instance, deploy a team of robots to guide and escort a group of people, using virtual attractive and repulsive forces and no explicit guidance signals. This kind of application, however, differs from most guide robots applied to public spaces commonly found in Literature. The presented team of mobile robots does not make any effort to appear more sociable. As the authors themselves point out, the application is more akin to shepherd dogs guiding a herd of sheep: we will therefore refer to this kind of robots as *shepherd* robots to distinguish them from *guide* robots.

Clodic et al. [10] and later Pandey and Alami [12] developed a guide robot which understands the commitment and intentions of the human. When people follow a robot guide, they may not always stay behind it, but sometimes deviate or stop temporarily. Rakham uses trajectory prediction to assess whether there is a change in the human's path and intentions and to provide fluid, natural and socially acceptable motion near the human.

More recently, Zhang [34] proposes the use of artificial potential fields to adapt the robot's motion to the uncooperative behaviors of human followers. Fiore et al. [35] describe a framework that assesses the quality of commitment of the user and decides whether to adapt the robot's speed, to suspend the task or to abandon it.

As we can see, recent research focuses on adapting the robot's motion to the user's behavior. In order to do so, whenever a human and a robot agent cooperate to perform a joint activity, it is necessary to model said cooperation and the level of commitment of the agents to the shared task.



(a) Rhino (b) Spencer

FIGURE 2.3: Two examples of guide robots. Left: Rhino (1998) [7] (Source: `www.researchgate.net`). Right: Spencer (2016) [19] (Source: `www.twente.com`).

2.3 Joint Activities and Cooperation

As previously mentioned, service robots are meant to help and assist humans, but the way they do may vary. Some tasks may require the robot to cooperate with the human, especially when the user or the robot may not achieve the ask alone. Following [36], we define three types of cooperation:

- **Assistance**: a robot may assist a person trough guidance. Typically, it consists of spoken instructions. For example, describing the steps for completing a task, or reminding a person his agenda and schedule fall into this category. Personal assistants and cognitive assistants that help people with dementia, such as in [18], provide assistance to humans. The robot helps the human without completing the task on his behalf.

- **Co-working**: two agents may cooperate by working in parallel on two different subsets of the shared task. Consider the example of a human and a robot gathering objects scattered in a room, as in [37]: both human and robot may pick objects on their own, but the robot should not interfere with the human's work. Therefore, it must understand and even predict which object the human will pick up, and take into account his preferences. In this type of cooperation, however, any agent could complete the task without the help of others.

- **Collaboration**: this is the category where the robot and human act jointly on the same object, space, or target. Typically it consists of a physical task, such as handing over an object, lifting it together, and so on. Guiding a person towards a

destination would fall into this category, since the cooperation of both the guide and guided agents is necessary for the task completion.

In this thesis, we focus on the collaboration case. In this type of cooperation, acting individually is not sufficient to complete the task: agents need to act *jointly*. In a joint action, participants share the same goal and a common plan of execution. Collaborative plans cannot be reduced to the sum of individual plans, but they consist of an interplay of agent actions [38]. A shared plan is always present whenever human-robot collaboration is performed, even if only implicitly.

Several approaches have been proposed to formally model agent cooperation and to implement joint execution of shared plans. Bütepage and Kragic in [39] provide a detailed review and description of mechanisms that arise in human-robot cooperations, as well as expressing them with a probabilistic framework called Sensorimotor contingency.

2.3.1 Joint Intention Theory

Joint Intention (JI) Theory [40] [41], is an extension to the BDI architecture. BDI (Belief, Desire, Intention) [42] is a popular architecture that models rational agents with a *belief* set, a *desire* set and an *intention* set. Beliefs are the information that the agent has about the state of the world. Desires are world states that the agent would ideally achieve. Intentions are desires that the agent has committed to bring about.

While BDI can be used to program intelligent agents and several implementations of this model have been developed, such as SPARK [43], AgentSpeak [44] and Procedural Reasoning System [45], it is not expressive enough to model the mechanisms and interactions of agent cooperation. The Joint Intention formalism, instead, is capable of defining the mental states assumed by agents during teamwork. Cohen and Levesque describe Joint Intention as a joint commitment to perform a collective action while in a certain shared mental state. It is what binds the team members together and makes the difference between a functional team and a disfunctional one. In other words, two agents may be cooperating, but with a shared mental state, they cooperate *effectively*.

More specifically, two agents are said to jointly intend to do an action (or actions) a if they have a joint commitment to doing the action a mutually believing throughout the

action execution that they are jointly doing that action as a team[40]. Therefore, while performing a collaborative action, the agent should ascertain that such mutual belief is ensured for all other agents.

Joint Intention Theory is defined formally using a modal language based on first order logic, with the addition of temporal operators and propositional attitudes. For a complete description of all mental state definitions, we refer to [41]. This formalism has already been used in robotic applications to improve the cooperation level between heterogeneous agents, as in [46] and [47].

2.3.2 Inferring Human Intentions

When cooperating together, the human and the robot do not perceive the shared task and shared environment in the same way, and they may have a different knowledge about the state of the world. Their beliefs about the world may differ, and such incomprehension may severely hinder the completion of the task. More so, the robot usually does not know how the human plans to achieve the common goal. Both participants only have a partial knowledge about the shared plan. Several research works have investigated how to estimate and infer the human's intentions.

In [39] two levels of prediction are presented:

- **Low-level prediction** is applied to immediate sensory changes caused by human movements, and is commonly performed trough Kalman filters. Extrapolation of the human's speed, previous positions and other features is used to predict the human's trajectory and ensure a socially acceptable navigation in [48] and [34]. Hoeller et al [49] uses potential fields to predict human trajectories.
- **High-level prediction** estimates the human's intentions and predicts his future behavior. Koppula and Saxena [50] infers human intentions trough object affordances: if the human approaches a region where he can pick an object, he probably intends to pick the object. Liu and Wang [51] use Finite State Machines to model human behaviors and predict them. Their main contribution is the capability to account for humans *undoing* actions. Koay et al. [23] stress the fact that inference works both ways: the human should be able to understand and predict the

robot's actions. Karami et al. [36], instead infers the user's intentions by learning Human-MDP (Markov Decision Processes) models. A similar approach is adopted in [52] as well. MDPs are described in detail in Section 3.3.1.

2.3.3 Human Attention

The attention of a person relates to the entity which is currently his focus of interest, or lack thereof. Head and gaze direction is a major cue for understanding where the attention of a robot or human is currently focused on. Therefore it can be used to understand if a person is currently addressing the robot [53] and, on the other hand, to allow a robot to shift its attention to its speaker [54]. Sisbot et al. [55] use gaze detection to estimate human attention, but they are more specific. They describe the distinction between the Field of View (FOV) of a person, that is, what the person sees, and its subset, the Field of Attention (FOA), which is what the person is currently *looking at*. Hoque et al.[56] show how a robot's head and eye movements can be used to control the attention of a person.

Human attention is closely tied to human intention. Kopp and Gärdenfors [57] show how attention can be considered as the first level of intentionality. Lallée et al. [29], describe how a robot can use head and gaze direction to show its current attention and allow users to predict its intentions.

The difference between intention and attention is that human attention may not have a specific target and may not be related to a task. Efforts in intention prediction are usually constrained within the scope of a shared task. Rather, a person may be looking "elsewhere", with no other intention than the looking itself. Even while performing a shared task, users may have an unfocused behavior, be distracted, or even change their mind and abandon the task. This is especially true in service applications in public spaces, where users are often customers passing by or bystanders. In such applications, naive users with no prior training that happen to accidentally pass by a robot may leave the interaction at any moment. For example, Pitsch et al. [2] adopt a "pause and restart" dialogue method to get the attention of users and engage in the interaction.

2.4 Chapter Conclusions

HRI applications, and specifically service robotics in public spaces, is a vast domain where research develops into many directions. Our work does not focus on developing advanced cognitive models for the sociability of the robot, hence it falls within the category of *socially situated* robots. The guide robot used during the thesis, detailed in Chapter 5, uses Proxemics for both maintaining a socially acceptable distance with the human trough the guidance task, and for estimating his level of cooperation. While some of the reviewed works use potential fields to implement the guidance task, our work uses an AI approach and decision-making models (described in Section 3.3). Despite not implementing it formally, we use Joint Intention Theory as a source of inspiration for modeling human-robot cooperation. More specifically, we focus on estimating the current level of attention of the human partner during a shared task, and accounting for it within the planning module and trough the whole execution of the guidance task. While several studies have been carried out to detect and control the user's attention, most do so to infer the person's intentions, or within a dialogue context, or only do to start an interaction. Few researches account for human attention within a planning context trough the whole shared task.

Chapter 3

Planning Under Uncertainty

In this Chapter, we review the main mathematical models adopted for planning a task. We describe both deterministic and probabilistic models. However, as mentioned in Section 2.2.1, the domain of this thesis provides several constraints, such as the unpredictability and uncertainty of the dynamic environment and of the human-robot cooperation. These constraints call for planning models capable of dealing with them. Hence, we will focus on probabilistic planning, and more specifically, probabilistic planning under uncertainty.

3.1 Overview of planning

Planning is the process of finding a plan, that is, a sequence of actions performed by an agent, capable of bringing the system to a desired goal state. We define a *system* as a couple consisting of an environment and of a set of agents situated in it.

A *state* is a description of a possible configuration of the system. The state is usually defined by a set of variables whose values may change over time. A system may be closed or open: a closed system cannot be in a state that does not belong to the fixed set of states modeling the system. An open system, on the contrary, may be in a state that cannot be identified by the set of variables. Models may also differentiate themselves depending on whether the state variables are continuous or discrete.

An agent affects the environment of the system through actions. When an agent performs an action, it changes the state of the system.

A *transition* is the passage from one state to another.
It is defined as a tuple $< starting\_state, action, final\_state >$. Transitions may be deterministic or probabilistic. A deterministic transition always leads to the same final state for a given action performed in a given starting state. A probabilistic transition may lead to a set of output states, and is associated with a probability distribution over said set. Trivially, deterministic transitions can be expressed as probabilistic transitions with a probability of 1.
As the system evolves through time, a transition occurs at each time step. The model is said to be stationary if the transitions do not depend on time.

The state of the system may be fully or partially observable. A system is said to be *partially observable* when the current state of the system is not always known. This is often the case in several problems, especially in real-world applications where sensors are subject to noise and errors, and may only provide information to the agent with limited accuracy, and when perception systems are limited by occlusions, blind spots, limited range and similar problems. The different types of planning can be summarized in Table 3.1.

	Full Observability	Partial Observability
Deterministic	Classical Planning	PKS
Stochastic	MDP	POMDP

Table 3.1: Planning models

3.2 Classical Planning

Classical planning is the simplest form of planning, since it is constrained by eight assumptions [58]:

1. *Finite System*: the system has a finite set of states and actions
2. *Fully observable*: the agent has full knowledge about the state of the system
3. *Deterministic*: all transitions are deterministic

4. *Static*: only the agent's actions can change the state of the system, and no exogenous event can occur
5. *Restricted goals*: the agent has no other goal than to reach a set of goal states
6. *Sequential plans*: a plan solving the problem is a linearly ordered sequence of actions
7. *Implicit time*: actions and events are instantaneous state transitions, they have no duration
8. *Offline planning*: dynamic changes occurring during the planning process are not accounted for by the planner

One of the most popular classical planner is STRIPS [59] (STanford Research Institute Problem Solver). The language of STRIPS is based on predicates, also called well-formed formulas. *States* are described as sets of predicates. For example, `IN(box1, room1)` $\wedge$ `PUSHABLE(box1)` describes a world state where box 1 is in room 1 and it can be pushed. STRIPS works on the assumption that anything not explicitly stated is false. *Actions*, or operators, consist of a set of preconditions that must be met to use the operator, and a set of postconditions that describe its effects. Postconditions are predicates that become true when the action is performed, therefore enabling a *transition* from one state to another. For example, the operator `GOTO(room1, room2)` that allows a robot to go from room 1 to room 2, is defined with `IN(robot, room1)` as precondition and `IN(robot, room2)` as postcondition.

A STRIPS problem can hence be described formally with a tuple $< P, O, I, G >$ where

- P is a set of predicates
- O is a set of operators
- I is an initial state
- G is a goal state

STRIPS problems are solved using a theorem prover, that checks if the goal conditions are met in the current state: if not, it looks at the differences between the current and the goal state, and chooses an operator to reduce these differences. The chosen operator's

preconditions are treated as a sub-goal to be reached, so another operator is chosen to satisfy the preconditions, and so on. Since STRIPS is a linear solver, however, it tries to satisfy one goal condition completely before dealing with other goal conditions, and it cannot undo a satisfied goal. It has been proved [60] that there exist simple problems that cannot be solved using this method. For solving this category of problems, it is required to undo a sub-goal to reach another sub-goal, a phenomenon known as Sussman Anomaly.

Partial order planners [61] have been developed to overcome such anomaly. They are based on the *least commitment* principle: that is, the idea that decisions should be deferred as long as possible. Specifically, deciding the order of actions should be performed only when necessary, and the planning process should reason about partial orders of actions. As an example, buying several ingredients is a requirement for cooking, and going to the store is a requirement for buying ingredients, but the order with which the individual ingredients are bought is irrelevant, as long as all of them are bought before starting cooking.

Petrick and Bacchus [62] have developed a framework called PKS (Planning with Knowledge and Sensing), capable of constructing plans with partial observability. It is based on a generalization of STRIPS. It replaces the sets of predicates that describe the state of the system with databases that represent the agent's knowledge about the state of the system. Therefore, actions are modeled as updates of the agent's knowledge rather than updates of the real world state.

Another popular formalism for classical planning is PDDL (Planning Domain Definition Language) [63]. PDDL is meant to be a unified language for modeling planning domains, inspired by several previous formalisms. As such, no specific planning algorithm is presented, but the language is compatible with several planners. While it uses a different syntax, PDDL shares a similar structure with STRIPS.

Despite the possible extensions to classical planners, the *Deterministic* assumption severly limits their use in real-world applications. This is especially true when operating in dynamic and environments and when cooperating with the unpredictable human behavior, as described in Section 2.2.1. The uncertainty about the evolution of the world state and about the human's reaction to robot's actions can be more easily handled in probabilistic terms.

3.3 Probabilistic Planning: Markov Decision Processes

Probabilistic planning accounts for both the uncertainty of the environment and the uncertainty of an action's effect. Actions taken by the agent may succeed, fail partially or fail totally, and the environment may evolve autonomously in a probabilistic way.

Consider the following example. A light-weight flying drone plans its own trajectory towards a destination in strong windy weather conditions. When the drone plans to move north, there is a probability that the wind pushes the drone north-west or west instead, so that the next position of the drone is not deterministic. In addition, the wind may change direction with a given probability. Hence, the direction of the wind is taken into account as a state variable whose evolution cannot be controlled by the drone's actions but can be estimated in terms of probabilities.

The given example is fully observable: the position of the drone is not deterministic but known. For an example of a partially observable setting, consider that the drone uses a GPS system to track its own position. The GPS system, however, may be unreliable and not sufficiently accurate to track the exact position on a small scale. In this case, the drone has to cope not only with the uncertainty on the next position (caused by the wind), but also with the uncertainty on its current position.

3.3.1 Definition of Markov Decision Processes

Markov Decision Processes (MDP) [64] are an efficient framework for planning under probabilistic constraints.

***Definition** (MDP)* 3.1. A Markov Decision Process is a controlled stochastic process, defined as a tuple $< S, A, T, R, H >$, where:

- S is a discrete and finite set of states s;
- A is a discrete and finite set of actions a;
- $T : S \times A \mapsto \Pi(S)$ is a transition probability function, such that $T(s, a, s') = Pr(s'|s, a)$;
- $R : S \times S \times A \mapsto \Re$ is a reward function;

- H is the planning horizon;

The transition function $T(s, a, s') = Pr(s'|s, a)$ gives the probability to move to state s' when action a is performed in state s. In a MDP, the Transition function satisfies the Markov property [65] that is, at a given time step t, the probability to reach a state s_{t+1} after performing action a_t in state s_t depends neither on the history of previous states $s_0, ..., s_{t-1}$ nor on the history of previous actions $a_0, ..., a_{t-1}$. In other words,

$$Pr(s_{t+1}|s_0, a_0, s_1, a_1, ..., s_t, a_t) = Pr(s_{t+1}|s_t, a_t)$$

The reward function $R(s, a, s')$ assigns a reward or cost whenever the agent performs an action a in state s resulting in state s'; while this is the most general definition, the reward function is commonly defined without the final state, as $R(s, a)$, or solely over the state, as $R(s)$. The reward function allows to define the goal states by assigning a great reward to them. Depending on the application, costs may be used to model the effort that it takes to perform the action, or to model negative consequences that it may have and that are not modeled in the system. Consider the example of an outdoor mobile robot. The robot is able to navigate trough rough terrain to reach a goal destination, but doing so requires more effort and may wear out the robot or even damage it, while navigating trough roads is safer. In the case of an autonomous vehicle, it may be uncomfortable for the passengers, even if the passengers' comfort is not modeled as a state variable of the system. If the shortest path leading to the destination is trough the rough terrain, the robot has to evaluate the cost of taking that path and whether or not it is "worth it". Therefore, the reward function introduces efficiency criteria in the decision-making model: the planning process will attempt to find the optimal plan with respect to the costs and rewards associated with each action at each time step.

The horizon H is the number of actions the agent will take during its life time. In an MDP, the planning process consists in finding an optimal policy π^*. A *policy* is a strategy used by the agent to reach a goal state. It tells the agent which action to perform given the current state of the system. Policies may have a finite, infinite or indefinite horizon. Finite-horizon policies only define the optimal action to take for a limited number of time steps, which may not be sufficient to reach a goal state from the initial state. Infinite-horizon policies do not terminate after a fixed number of time

steps. The system runs indefinitely while trying to maximize the reward. The planning process for infinite-horizon policies terminates when the optimal policy is found, within a precision range ϵ. Indefinite policies have a finite but unknown number of time steps. The planning process runs until a goal state is reached.

In a stationary, infinite horizon MDP, the policy consists of a list of state-action pairs. The optimal policy maximizes a value function $V^\pi : S \to \Re$, which associates to each state $s \in S$ the expected total reward (also called *value*) gained when applying policy π from that state.

The specific definition of the value function depends on the choice of the performance criterion that should be optimized. Given r_t as the reward obtained at time t, the performance criteria and the corresponding value function definitions are the following:

1. The finite horizon criterion: only maximizes the expected gain up to a given horizon H.
$$V_H^\pi(s) = E\left[\sum_{t=0}^{H-1} r_t\right] \quad \forall s \in S$$

2. The γ-discounted finite horizon criterion:
$$V^\pi(s) = E\left[\sum_{t=0}^{H} \gamma^t r_t\right] \quad \forall s \in S$$
The γ parameter allows to tune the "greediness" of the solution: the lower the value, the less weight rewards at future time steps will have with respect to immediate rewards.

3. The γ-discounted infinite horizon criterion: equivalent to the discounted criterion with $H = \infty$.
$$V^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \quad \forall s \in S$$

4. The total reward criterion: equivalent to the discounted infinite horizon criterion with $\gamma = 1$.
$$V^\pi(s) = E\left[\sum_{t=0}^{\infty} r_t\right] \quad \forall s \in S$$

5. The average criterion:

$$V^{\pi}(s) = \lim_{n\to\infty} E\left[\frac{1}{n}\sum_{t=0}^{\infty} r_t\right] \quad \forall s \in S$$

For infinite horizon problems, the γ-discounted criterion is the most commonly used. The Bellman equation [66] shows how to compute the value function in a recursive way:

$$V_t(s) = \max_{a\in A}\left[R(s,a) + \gamma \sum_{s'\in S} T(s,a,s')V_{t-1}(s')\right] \tag{3.1}$$

3.3.2 Value Iteration

Value Iteration [66][67][64] is the most commonly used algorithm for solving MDPs. It directly applies the Bellman equation to iteratively compute the optimal value function V^*. It starts with an arbitrary value function and then refines it at each iteration step by finding the best value for all possible actions and states. For a finite horizon problem, this loop ends when it reaches the maximum horizon step. For infinite horizon problems, the refinement process is terminated when the value does not improve anymore (within a threshold ϵ).

The algorithm is described in Algorithm 1. It first initializes the value function with an arbitrary value. Then, it applies Equation 3.1 at each iteration step for all states. This process terminates when the value improvement is below a given threshold ϵ. For finite horizon problems, it may also terminate after a fixed number of iterations. Then the algorithm returns the policy generated by associating to each state the action that gave the highest value. The complexity of each iteration of the the algorithm is $\mathcal{O}(|S^2||A|)$ [68].

3.3.3 Policy Iteration

Policy Iteration [69] is another commonly used algorithm for solving Markov Decision Processes. It is described in Algorithm 2, assuming a discounted criterion. It starts with an arbitrary policy and improves it at each iteration. To do so, it computes the value function of the policy π_t as a set of $|S|$ linear equations in $|S|$ unknown variables, that

Data: S, A, T, R, H, ϵ
Result: optimal policy π^*
Assign V_0 arbitrarily $\forall s \in S$ $t \leftarrow 0$ **while** $\max_{s \in S}(|V_t(s) - V_{t-1}(s)|) < \epsilon and t < H$ **do**
  forall $s \in S$ **do**
    $$V_t(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s') \right]$$
  end
  $t \leftarrow t + 1$
end
forall $s \in S$ **do**
  $$\pi^*(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_t(s') \right]$$
end
return π^*

Algorithm 1: Value Iteration

is, the values $V^{\pi_t}(s)$ for each state $s \in S$. The algorithm stops when it cannot improve the policy anymore. The complexity of the algorithm is $\mathcal{O}(|S^2||A|) + \mathcal{O}(|S^3|)$ [68].

Data: S, A, T, R
Result: optimal policy π^* and the associated $V*$
Assign π_0 arbitrarily $t \leftarrow 0$ **while** $\pi_t! = \pi_{t+1}$ **do**
  Solve
  $$V_t(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s') V_t(s') \quad \forall s \in S$$
  forall $s \in S$ **do**
    $$\pi_{t+1}(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_t(s') \right]$$
  end
  $$t \leftarrow t + 1$$
end
return V_t, π_{n+1}

Algorithm 2: Policy Iteration

3.3.4 Factored Markov Decision Processes

In small-scale MDP problems, the state-space of the system is commonly represented as an enumeration of all the possible states. This representation becomes more complex and cumbersome in large-scale problems, since the size of the state-space increases exponentially with the number of features included when modeling the problem.

Factored Markov Decision Processes (FMDP) [70] [71] are an extension to MDPs that exploit structure in the problem in order to allow a more compact representation. In a FMDP, the state space is generated by the product of discrete state variables.
Let $x_1, ..., x_n$ be the state variables of the model. Let $Dom(x_i)$ denote the domain of the i-th state variable. The current state of the model can be represented as a vector of instances of all state variables: $\mathbf{x} =< x_1, ..., x_n >$, and the factored state-space of the FMDP can be generated as $S = Dom(x_1) \times ... \times Dom(x_n)$.

A major advantage of using FMDPs is the possibility to exploit dependencies and independencies of state variables in the transition probabilities. For example, a state variable x_i may only depend on its previous value, regardless of the value of other variables: $Pr(x_i'|x_1, ..., x_n) = Pr(x_i'|x_i)$.

The dependency between state variables can be represented as a Dynamic Bayesian Network (DBN) [72]. A Bayesian Network is a directed acyclic graph where nodes represent variables and edges represent dependencies. DBNs extend Bayesian Networks with temporal information: nodes represent the state of a variable within a time step t. Figure 3.1 shows an example of DBN: the evolution of the *Weather* (e.g. Rainy or Sunny) variable through time depends only on its previous state; instead, a person's *Activity* (e.g. DoingHomework, WatchTV or GoingToBeach) is influenced by both the state of the *Weather* and the previous activity (for example, the person may finish homework or get bored with its previous activity).

3.3.5 Hierarchical Markov Decision Processes

When using MDPs to model real-world problems, the size of the state-space may quickly increase in size. Since the complexity of MDP resolution algorithms is polynomial with respect to the number of states, solving large-scale models may become intractable. An efficient approach for planning large problems is to use a hierarchical decomposition.

No formal and global definition of Hierarchical Markov Decision Processes (HMDP) is given, since several different approaches have been developed.

Dean and Lin [73] introduced region-based hierarchical decomposition for MDPs. It partitions the state-space into several regions, arguing that not all state variables are relevant in all regions of the state space. Therefore, the original MDP is decomposed

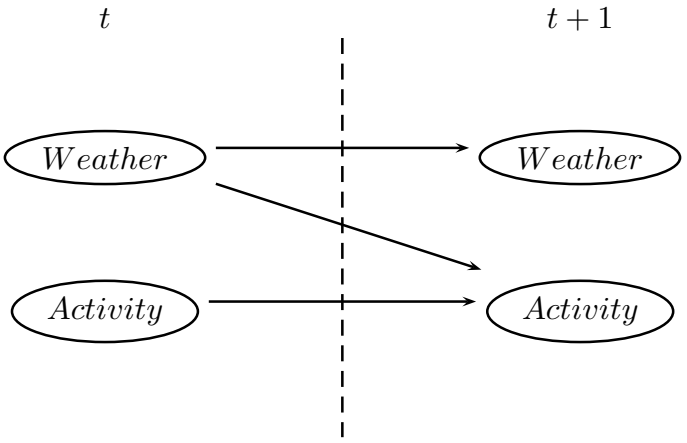


FIGURE 3.1: Example of a Dynamic Bayesian Network.

into smaller MDPs, each of which provides a local policy. To combine the local policies, an abstract MDP is constructed, which considers each region as an abstract state and each local policy as an abstract action. Abstract actions are also called *macro-actions* [74].

Similarly, Dearden and Boutiier [75] presented an abstraction method that ignores the less relevant state variables to reduce the size of the state-space. The generated abstract policy, however, does not simply guide and accelerate the search for a solution at less abstract levels, but can be executed directly. Thus, the abstract policy acts like an approximate optimal policy instead of a compact representation of the complete policy.

The Options framework, instead, builds a temporal hierarchy for MDPs [76]. Options extend the MDP framework and allow the definition of actions on different time scales. Options are quite different from macro-actions: while macro-actions still consist of single-step transitions between abstract states, not unlike primitive actions, options are temporally extended and may take a variable and unknown amount of time to be executed.

Other methods aggregate states that behave similarly, such as [77]. A review of state abstraction methods is given by Li et. al. [78]. Other approaches, such as MAXQ [79] and PolCA [80] perform a hierarchical decomposition based on actions: the original task is decomposed into smaller sub-tasks easier to solve.

Hierarchical decomposition is closely related to the principle of knowledge reuse. Once a local policy is computed for a region of states, the same policy can be used again for similar regions [81].

3.4 Partial Observability with MDPs

Since Markov Decision Processes rely on the assumption that the state is fully observable, they cannot be used for planning in partially observable problems. On the other hand, stochastic processes with partial observability have been modeled as Hidden Markov Models (HMMs) [82], which are a subset of DBNs. In a HMM, observations are generated with a given probability whenever a state transition is performed. For a given sequence of observations, it is possible to compute the most likely sequence of states that has generated it and, more specifically, compute an estimate of the current state of the system. HMMs, however, are not controlled processes, since there is no action performed by agents.

	Full Observability	Partial Observability
No Control	Stochastic Process	HMM
Controlled	MDP	POMDP

TABLE 3.2: Controlled processeses and Partial Observability

3.4.1 Definition of POMDPs

Partially Observable Markov Decision Processes (POMDPs)[83] [84] are an extension to MDPs that take into account the partial observability of the system. The planning agent receives observations from the system with a given probability, allowing it to keep an estimate of the current state. This estimate is a probability distribution over the state-space, called *belief state*.

***Definition** (POMDP)* 3.1. A Partially Observable Markov Decision Processe is a tuple $< S, A, T, \Omega, R, O, H >$, where:

- S is a discrete and finite set of states s;

- A is a discrete and finite set of actions a;
- $T : S \times A \mapsto \Pi(S)$ is a transition probability function;
- Ω is a discrete and finite set of observations o;
- $R : S \times S \times A \mapsto \Re$ is a reward function;
- $O : S \times S \times A \mapsto \Pi(\Omega)$ is an observation probability function.
- H is the planning horizon.

The observation function $O(s, a, s', o) = Pr(o|a, s, s')$ gives the probability of observing o when performing action a in state s results in arriving into state s'. In most applications found in Literature the observation function is defined solely over the actions and final states as $O(a, s', o)$ [1]

At any time step t, the agent keeps a belief state b to estimate its current state:

$$b_t(s) = Pr(s_t = s)$$

The belief state is a sufficient information state [85] that sums up the information received from observations until the current time step. Therefore, a POMDP satisfies the Markov property and no complete history of observations is required.

Whenever the agent performs an action a and receives an observation o, it updates its belief state according to the new information received and to the predicted transition given by the transition function. This belief update is computed in the following way[86]:

$$\tilde{b}_o^a(s') = \frac{\sum_s O(s, a, s', o) T(s, a, s') b(s)}{\sum_{s'} \sum_s O(a, s', o) T(s, a, s') b(s)} \quad \forall s' \in S \tag{3.2}$$

$\tilde{b}_o^a$ is hence the resulting belief when executing action a in belief state b and observing o.

The belief update function can be considered as a transition function between beliefs. Actually, a POMDP can be modeled as a special MDP, called Belief-MDP (BMDP)[84], where we consider each belief b as if it were a state of an MDP.

[1] The observation function can also be defined without any loss of generality over the starting states rather than the ending state: $O(s, a, o)$ is the probability of observing o when performing action a from state s.

Definition *(BMDP)* 3.1. A Belief-MDP is a POMDP modeled as a tuple $< \mathcal{B}, A, t, r >$, where:

- $\mathcal{B}$ is the continuous set of belief states b;
- A is a discrete and finite set of actions a;
- $t(b, a, b')$ is the belief state transition function;
- $r(b, a)$ is the reward function on belief states.

The belief transition function $t(b, a, b')$ can be defined as:

$$t(b, a, b') = P(b'|b, a) = \sum_{o \in \Omega} P(o|b, a)\delta(b', \tilde{b}_o^a)$$

Where $P(o|b, a)$ is the conditional probability of an observation

$$P(o|b, a) = \sum_{s \in S} \sum_{s' \in S} O(a, s', o)T(s, a, s')b(s)$$

and

$$\delta(x, y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases}$$

The reward function $r(b, a)$ is computed from the POMDP reward function as:

$$r(b, a) = \sum_{s \in S} R(s, a)b(s)$$

As for MDPs, we can apply the Bellman equation to BMDPs as well:

$$V_t(b) = \max_{a \in A} \left[\sum_{s \in S} R(s, a)b(s) + \gamma \sum_{o \in \Omega} P(o|b, a)V_{t-1}(\tilde{b}_o^a) \right] \tag{3.3}$$

3.4.2 Planning with POMDPs

Despite the possibility to model a POMDP as a Belief-MDP, it is impossible to use Value Iteration to compute policies because of the continuous nature of the belief space $\mathcal{B}$ which acts as the BMDP's state-space.

However, the value function of POMDPs shows special properties that can be used to build a tractable and efficient algorithm. It has been shown [83] that the optimal value function V^* for a finite horizon problem is piecewise linear and convex (Figure 3.4.2). Specifically, V^* is the upper surface of a set of hyperplanes through the belief-space, where each hyperplane is the value function of a possible policy. Each linear segment is represented by a vector of S coefficients, called $\alpha-$vector, which describes the equation of the hyperplane.

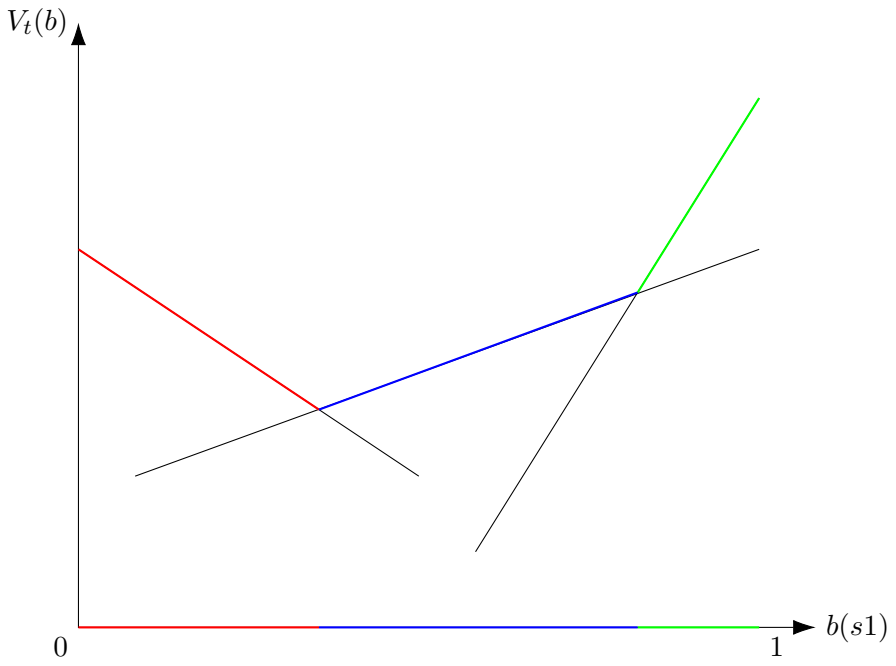


FIGURE 3.2: Piecewise linear convex Value function. In this example, the POMDP model has two states, and $b(s2) = 1 - b(s1)$.

For a horizon 1 problem, each vector corresponds to one action (since each policy contains exactly one action), therefore the $\alpha-$vectors trivially represent the immediate reward obtained when executing said action at a given belief point. The optimal policy then associates the best action to regions of the belief-space. For a higher horizon $n > 0$, the Value function V_n can be computed by taking into account the value functions associated with every possible action and observation. For a given action a, it is possible

to compute $V_n^{a,o} \quad \forall o \in O$, that is, the value of policy of horizon n starting with action a and proceeding with the optimal horizon $n-1$ policy taken after observing observation o. Then, it is possible to use these $|O|$ value functions to calculate the value function of V_n^a, that is, the value of policy of horizon n starting with action a [87]. This step is performed for every action $a \in A$, so that $|A|$ value functions can be combined to compute V_n for horizon n.

Some of the vectors generated may be dominated by others. A vector θ is dominated if $\forall b \quad \exists \theta' \quad | \quad V^{\theta}(b) < V^{\theta'}(b)$. Dominated vectors are useless for determining the optimal value function $V^*(b)$ and thus can be pruned. Most POMDP resolution techniques focus on how to detect and prune such dominated vectors.

$\alpha-$vectors have been used to adapt Value Iteration to POMDPs and find optimal solutions. Exact algorithms have been developed by Sondik [88], Littmann [89] and Cassandra et al. [90][86]. The Witness algorithm [89] attempts at pruning dominated vectors by defining regions in the belief space for each vector and looking for a point where the vector is not dominant. The algorithm generates and maintains a collection of policy trees, called Q-functions Q_t^a; the function $Q_t^a(b)$ gives the expected reward for taking action a from belief b and then acting optimally for the remaining $t-1$ steps, and is defined in the following way:

$$Q_t^a(b) = \sum_s b(s)R(s,a) + \gamma \sum_o Pr(o|b,a)V_{t-1}(\tilde{b}_o^a)$$

with $V_t(b) = max_a Q_t^a(b)$, as per in Value Iteration. The algorithm starts by generating the vector (that is, a policy tree) from an arbitrary belief state and adding it to a set $\tilde{\mathcal{Q}}_t^a$ of non-dominated vectors. At each iteration step, it looks for a "witness" belief state b that can testify the fact that the set $\tilde{\mathcal{Q}}_t^a$ is not yet a perfect representation of the desired $Q_t^a(b)$. To do so, we define regions for each vector where that vector is assured to be dominant. If it is possible to find a belief point where a different strategy would perform better, then we can use it to find the vector missing from $\tilde{\mathcal{Q}}_t^a$ and add it to the set.

Despite their efficiency, exact algorithms require an exponential number of elements to represent the value function. As such, they can only be applied to toy problems, with only a few states, and cannot be applied to real-world, large-scale problems. To overcome such issue, approximate algorithms have been developed.

The basic idea of approximate Value Iteration algorithms is to consider only a finite set $B \in \mathcal{B}$ of belief points. The possibility to compute successful policies for much larger problems, however, comes at the cost of optimality. Lovejoy [91] first proposed this approach using a regular finite grid of belief points. Pineau et al. [92] improved the algorithm arguing that some belief points are less likely to be reached by the POMDP, so it is unnecessary to treat all beliefs equally as in regular grids. Therefore, the proposed Point-Based Value Iteration algorithm uses action simulation to expand the belief set. Other approaches use random exploration [93] or heuristic search [94] in order to build iteratively their belief set.

The result of POMDP planning algorithms is a policy usually represented as a tree. A POMDP policy tree consists of nodes representing actions and edges representing observations. The depth of the tree is equal to the policy's horizon H.

3.4.3 POMDP Extensions

Most extensions that have been developed for MDPs have been adapted to POMDPs as well. POMDPs can be modeled using a Factored representation [95] [96]. Approximate resolution techniques, such as heuristic search value iteration, have been improved to exploit the factored model [97].

Even more so than MDPs, POMDPs suffer greatly from the exponential growth of the state-space in real-world applications. Therefore, POMDPs benefit greatly from a hierarchical structure. Pineau et al. [98] have developed a hierarchical approach for POMDPs, that has been especially effective for robot navigation [99].

Worth of notice for the scope of the thesis is also the extension that adds elements of the BDI architecture to POMDPs [47]. The compatibility and equivalence between the two formalisms has been shown by Schut et al. [100].

3.5 Chapter Conclusions

In this chapter we have presented a review of the main frameworks for planning and decision-making. Given the dynamic, uncertain and unpredictable nature of the thesis application scenario, the guidance task in a public space, we have described more in

detail the probabilistic approach, presenting the Markov Decision Processes and the algorithms to solve them. We also have showed how such models can be improved trough a factored and a hierarchical approach.

We have then introduced Partially Observable Markov Decision Processes, an extension to MDPs to handle incomplete knowledge and hidden state variables. In our thesis, we have chosen to model the human-robot cooperation in the guidance task as a POMDP. The problem is intrinsically not fully observable. We consider the level of attention of the human to be a hidden variable, that can only be estimated trough observations. Other important features, such as the proxemic distance between user and robot, presented in Section 2.2.1, may not be fully observable since occlusions may prevent the sensors to detect the person. Additionally, the human's behavior cannot be controlled by the robot, only partially influenced. Therefore, we consider the human component as an independent, uncontrollable variable, that may evolve in a stochastic way. This consideration motivates the use of POMDPs with respect to a partially observable deterministic planning method.

Part III

Contributions

Chapter 4

The Cooperation POMDP

In this chapter we describe the main contribution of the thesis: the development of a planning framework to ensure the cooperation of the human user during a joint activity in a public space. The framework is meant to address in particular several challenges:

1. **Real-time execution**: the human-robot interaction should be fluid and natural, therefore the robot should have fast reaction times.
2. **Uncertainty on human's behavior**: we do not assume a persistent commitment of the human user to the joint task, and we consider his behavior as unpredictable. The mental state of the human teammate is intrinsically not observable, and can only be estimated by the robot.
3. **Uncertainty on environmental information**: public spaces are dynamic and noisy environments, and sensor data may be inaccurate or occluded. Therefore, the robot may not have a complete knowledge of the world state.
4. **Infinite horizon planning**: because the human agent may not be always cooperating with the robot, the execution time required to achieve the task is unknown. Finite horizon planning is seldom applied to large-scale, real-world applications, since the problem resolution would quickly become intractable as the planning horizon increases.
5. **Robustness**: the robot should be able to react quickly and efficiently to any unexpected errors and failures that may arise in the unpredictable application environment.

Point 2, 3 and 4 motivate our choice to use POMDPs to efficiently generate plans able to achieve a task within a dynamic, unpredictable and uncertain environment. In the course of this Chapter, we will describe the solutions adopted to satisfy the other constraints and objectives.

Using POMDPs, we can model the application domain by defining the state variables, actions and observations. Solving the POMDP would then provide a policy for achieving the task. In order to account for the level of engagement of a human teammate in a joint task, however, we need to introduce additional features. For instance, as mentioned in Section 2.3.3, gaze and head orientation are fundamental clues to understand the focus of attention of a person. In a naive approach, these values would be added to the state-space of the POMDP. The human's level of cooperation would thus be accounted for within the state of system and during the decision-making process. Such approach, however, would increase the complexity of the POMDP model, and policy generation would quickly become intractable.

4.1 Hierarchical framework

4.1.1 Overview of the framework

In order to avoid the problem of state-space explosion, we adopt two strategies: *state abstraction* and *task decomposition*.

- *State abstraction*: using an approach commonly adopted in Literature to reduce the complexity of POMDPs (Section 3.4.3), we perform an abstraction process which maps the application domain into clusters of states. The planning process is therefore performed on a reduced state-space and more efficient. This approach induces the hierarchical structure of our framework.
- *Task decomposition*: we divide the problem of human-robot collaboration in a shared task into its two fundamental aspects: the cooperation aspect, and the task to achieve itself. Each aspect is hence resolved separately, with a reduced state-space.

For the remainder of the thesis, we will refer to the human-robot collaboration in a joint activity in the following terms:

- The **Task** is the objective that needs to be reached, regardless of cooperation concerns. Handing over an object or reaching a destination are examples of tasks.
- The **Cooperation**, also called Joint Intention (JI), is a shared mental state where all agents in a team are committed to bring about a joint activity. We loosely draw inspiration from the Joint Intention Theory (Section 2.3.1) for describing the cooperative aspect of a joint activity.
- The **Mission** is the overall human-robot joint activity, where both agents have to collaborate to achieve a common objective. The Mission includes both Task and Cooperation aspects.

For instance, the Task aspect of a guide robot would consist in reaching the target destination, while the Cooperation aspect would consist in ensuring that the user is following the robot.

The approach proposed in this thesis therefore consists in a conceptual hierarchical framework, shown in Figure 4.1. The following sections of this Chapter will describe in detail the role of each module in the framework.

4.1.2 Hierarchical structure

The framework shown in Figure 4.1 consists of three layers, in an increasing level of abstraction: the *Primitive* layer at the bottom, the *Cooperation* layer at the middle, and the *Mission Status* layer at the top. Within the Cooperation layer, we define two sub-systems, each meant to perform planning on a subset of the Cooperation layer state-space. The main feature of our framework is the partition of the domain into two components: one module dealing with the task itself, and the other dealing with the cooperative aspects. Each module performs its own planning and decision-making.

At any time the robot first checks if cooperation with the user is ensured. If not, the cooperation module tries to re-establish the joint intention. Otherwise, the robot may

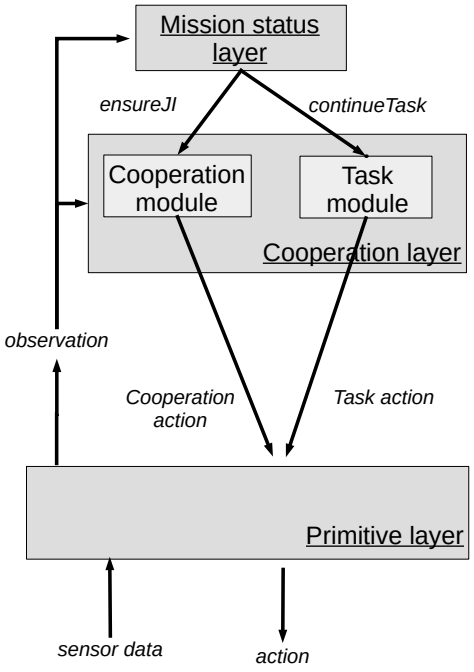


FIGURE 4.1: Structure of the framework

proceed with the task. A third, high-level module acts as a mediator between these two modules.

The hierarchical structure also provides a good degree of modularity. Each module of the framework can be implemented independently as long as their interfaces remain compatible and consistent with the general framework. This indipendence between modules allows our framework to be easily adopted in many HRI applications.

Primitive layer

The role of the Primitive level is to take information from sensors and execute low-level instructions, called *primitive actions*. It acts as an interface between the higher layers of the hierarchy, which perform planning at an abstract level, and the real application domain. In such a way, the framework is able to completely separate execution concerns from the general Mission planning process.

A conceptual role of this layer is to provide the low-level domain upon which higher level variables are defined through abstraction. Such a process is required for the computation of transition probabilities within the Mission Status and Cooperation layers POMDPs.

In order to keep the framework as general as possible, we do not model the Primitive layer in a strict and formal way. For the scope of the thesis, however, we represent it as a tuple $< S^{bot}, A^{bot}, T^{bot}, \Omega^{bot} >$, where:

- S^{bot} is the state-space of the application domain;
- A^{bot} is a set of primitive actions;
- $T^{bot} : S^{bot} \times A^{bot} \mapsto \Pi(S^{bot})$ is a transition probability function;
- Ω^{bot} is a discrete and finite set of observations;

Remark that we did not define the Primitive layer as a POMDP, since it does not have neither a reward function nor observation function. The Primitive layer does not necessarily need to perform any planning. The following mappings, however, must be defined [1]

- $S^{bot} \mapsto S^{mid}$
- $A^{bot} \mapsto A^{mid}$
- $\Omega^{bot} \mapsto \Omega^{mid}$

where S^{mid}, A^{mid}, Ω^{mid} are respectively the state-space, set of actions and the set of observations of the Cooperation layer. For instance, for a mobile robot, the low-level state-space would most likely include spatial coordinates, which would map into topological nodes or relative distances at higher levels. The action mapping $A^{bot} \mapsto A^{mid}$ is used to translate *macro-actions* from upper levels into sequences of *primitive* actions at the Primitive layer. We consider these action sequences to be modeled as Finite State Machines in order to be easily executable; details on action execution however are beyond the scope of this Chapter (an instance of implementation is described in Chapter 5).

[1] the surjectivity of these mappings is required

Cooperation layer

The Cooperation level generates plans of *macro-actions* for both achieving the task goal and re-establishing cooperation when needed. It consists of two sub-systems: the *Cooperation* and the *Task* systems.

We formalize the Cooperation layer as a tuple $< S^{mid}, A^{mid}, T^{mid}, \Omega^{mid} >$, where:

- S^{mid} is a discrete and finite set of states;
- A^{mid} is a discrete and finite set of macro-actions;
- $T^{mid} : S^{mid} \times A^{mid} \mapsto \Pi(S^{mid})$ is a transition probability function;
- Ω^{mid} is a discrete and finite set of observations;

We represent the state space of this layer in a factored way as the product of discrete state variables. We denote by $x_1, ..., x_n$ the state variables of the model, s.t. $S^{mid} = Dom(x_1) \times ... \times Dom(x_n)$. At the Cooperation layer, we define two sets of state variables: X^{task}, and X^{coop}.

- *Cooperation* variables $X^{coop} = (x_1, ..., x_c)$, with $c = |X^{coop}|$, are those variables which have been defined explicitly to deal with the Joint Intention problem and the Cooperation aspect of the Mission. These variables should formalize in an abstract way the relationship between the agents, such as spatial relationships, and the mental state of the human.
- *Task* variables $X^{task} = (x_{c+1}, ..., x_n)$ pertain to the Task itself, regardless of the human's level of commitment and mental state. These variables are *irrelevant* for establishing the status of cooperation among agents.

Whether cooperation with the human is ensured or not depends only on the current values of Cooperation variables $x_1, ..., x_c$. Therefore, by partitioning the domain of the Cooperation layer, we can define two separate and independent modules or sub-systems. Each module performs its own decision-making process pertaining to its own aspect of the Mission and may generate plans independently. The same reasoning can be applied to partition the set of actions A^{mid} and, eventually, the set of observations Ω^{mid} as well.

The Task module will provide a plan to reach a destination, hand over an object, and so on, while the Cooperation module will provide a plan to draw a person's attention, approach him, dialogue and so on. Such partition provides the framework with great flexibility: it allows each module to be modeled and implemented differently. More so, the policy generation for the Cooperation system may be performed offline and used for multiple, different instances of the human-robot collaboration task.

For the remainder of the thesis, we won't describe the model and implementation of the Task module and consider it as a black box. Instead, we focus on the Cooperation sub-system.

We model the Cooperation sub-system as a POMDP $< S^{co}, A^{co}, T^{co}, \Omega^{co}, R^{co}, O^{co} >$[2], where:

- S^{co} is the state-space generated by Cooperation variables $x_1, ..., x_c$
- $A^{co} \subseteq A^{mid}$
- $T^{co} : S^{co} \times A^{co} \mapsto \Pi(S^{co})$
- $\Omega^{co} \subseteq \Omega^{mid}$
- $O^{co} : S^{co} \times A^{co} \times S^{co} \mapsto \Pi(\Omega^{co})$

Mission Status layer

Both the Task and Cooperation modules generate a policy, which associates a macro-action a to each state $s \in S^{bot}$. The role of the Mission Status layer is to choose whether executing actions from the Task or Cooperation sub-system. In order to do so, it needs to check whether Joint Intention is ensured in the current state or not. If JI between human and robot is ensured, the robot may proceed with the task: the system activates the Task sub-system which will provide the action to be executed. Otherwise, the system will select the action from the Cooperation sub-system's policy to ensure cooperation.

Whether the JI is ensured or not only depends on Cooperative variables $x_1, ..., x_c$. Therefore, the domain of the Mission Status layer consists of the same Cooperative variables

[2]for the remainder of the thesis, we will omit the planning horizon H as we will consider only γ-discounted infinite horizon models, unless stated otherwise

of the Cooperation layer. The Mission Status layer can be modeled as a POMDP

$$< S^{co}, A^{ms}, T^{ms}, \Omega^{co}, R^{ms}, O^{ms} >$$

We define a set of Cooperative States $CS \subseteq S^{co}$ where the JI is preserved, that is, where all agents have a commitment to the common task and are actively performing actions to achieve it. As long as the current state $s \in CS$, then all agents are trying to achieve the common goal: everything is going well and there is no need to react. Otherwise, the agent needs to try to bring the system back to a CS state. In other words, CS is the set of states where no action is required specifically to repair the missing cooperation between human and robot. While the Mission Status layer shares the same state-space with the Cooperation sub-system, its set of actions consists of module activation actions, such as *continueTask* and *ensureJI*.

4.2 Policy generation

The flexible structure of our framework allows the Cooperation sub-system to be solved offline independently from the Task sub-system. As already discussed in Section 3.4.2, finding an optimal policy in the POMDP's continuous state space, however, is no trivial task.

The approach we have adopted is to represent the POMDP as a Belief-MDP (Definition 3.1) and discretize its belief-space $\mathcal{B}$. The set of beliefs $\mathcal{B}$ is continuous, and a discretization process is required in order to solve the BMDP with classic MDP algorithms (Section 3.3.2). The main advantage of this method is the possibility to easily generate infinite-horizon policies, and implement them on the real robot as Finite State Machines. In a POMDP, a policy consists of a tree of actions and observations. Such a policy can be implemented as a FSM, but an infinite horizon policy would require an infinite tree. By solving the POMDP as a discrete BMDP, we obtain an MDP-like policy, which is a list of state-action pairs. In a additional step, we reintroduce observations in the generated policy.

The offline planning process for the Cooperation POMDP module can be summarized in the following steps:

1. Formalize the Cooperation sub-system as a POMDP
2. Discretize the belief-space and obtain the associated Discrete-BMDP
3. Solve the Discrete-BMDP with standard MDP resolution algorithms
4. Translate the generated DBMDP policy into a POMDP policy
5. Implement and execute the POMDP policy as a FSM

4.2.1 Discretization method

In this section we describe how the belief space of the POMDP was discretized to generate the DBMDP. The most straightforward method to do so is to apply an uniform discretization factor on the whole belief space. The discretization factor k is the range of non-zero probability values that can constitute a probability distribution. For instance, with $k = 4$, probabilities can have the following values: $\{0, 0.25, 0.5, 0.75, 1\}$. The number of belief points that would be generated with an uniform discretization on the whole belief space, however increases dramatically with the number of states N and the discretization factor. It is computed as:

$$|B| = \binom{N+k-1}{k} = \frac{(N+k-1)!}{(N-1)!k!} \tag{4.1}$$

To reduce the complexity of the generated DBMDP, we suggest the following approach for the discretization of belief probabilities: instead of applying an uniform discretization on the POMDP's state space, we apply the discretization over the domain of the state variables, exploiting the POMDP's factored structure. In other words, instead of defining a set of discrete probability distributions $\Pi(S)$, we define sets of distributions $\Pi(Dom(x_1)), ..., \Pi(Dom(x_n))$. An uniform discretization factor may be used on the state variables' domains. We use the example given in Figure 3.1 to illustrate the approach. Given two variables x_1 and x_2 s.t. $Dom(x_1) = \{\text{Cloudy}, \text{Sunny}\}$,

$Dom(x_2) = \{\text{WatchTV}, \text{GoingToBeach}\}$ and a discretization factor $k = 2$, we obtain

$$\Pi(Dom(x_1)) \in \{[1,0],[0.5,0.5],[0,1]\}$$

$$\Pi(Dom(x_2)) \in \{[1,0],[0.5,0.5],[0,1]\}$$

Assuming the variables are independent, each belief state is computed as the product of probabilities for each variable to have the current state's values:

$$b(s) = Pr(x_1)Pr(x_2)...Pr(x_n)$$

where $s =< x_1, ..., x_n >$.

Similarly, each belief state can be represented as a vector of instances of probability distributions

$$b =< \Pi(Dom(x_1)), \Pi(Dom(x_2)), ..., \Pi(Dom(x_n)) >$$

The full discrete state-space of the resulting BMDP is then generated as the product of all probability distributions over the state variables' domains. Following the example, given $s_1 = (\text{Cloudy}, \text{WatchTV})$, $s_2 = (\text{Cloudy}, \text{GoingToBeach})$, $s_3 = (\text{Sunny}, \text{WatchTV})$, $s_4 = (\text{Sunny}, \text{GoingToBeach})$, and $\Pi_1(Dom(X)) = [1,0]$, $\Pi_2(Dom(X)) = [0.5,0.5]$ and $\Pi_3(Dom(X)) = [0,1]$ we get:

$b_1 = (\Pi_1(Dom(X_1)), \Pi_1(Dom(X_2))) = [1,0,0,0]$

$b_2 = (\Pi_1(Dom(X_1)), \Pi_2(Dom(X_2))) = [0.5,0.5,0,0]$

$b_3 = (\Pi_1(Dom(X_1)), \Pi_3(Dom(X_2))) = [0,1,0,0]$

$\vdots$

$b_5 = (\Pi_2(Dom(X_1)), \Pi_2(Dom(X_2))) = [0.25,0.25,0.25,0.25]$

$\vdots$

$b_9 = (\Pi_3(Dom(X_1)), \Pi_3(Dom(X_2))) = [0,0,0,1]$

Discretization of the continuous belief space introduces approximation errors. Using a smaller uniform discretization factor generates more accurate probability distributions, but it increases the resulting state space of the Belief-MDP. A trade-off is required between model accuracy and computational cost, that may quickly become unbearable as the model scales up. The advantage of the factored discretization is the possibility to

generate non-uniform distributions in a more flexible and customizable way. Each state variable may have a different discretization factor, or may take into account feature-specific constraints depending on the application domain.

In order to avoid ambiguity, given a Belief-MDP $< \mathcal{B}, A, t, r >$, we define the Discretized Belief-MDP in the following way:

Definition *(DBMDP)* 4.1. A Discrete-BMDP is a tuple $< B, A, \tau, \rho >$, where:

- $B \subseteq \mathcal{B}$ is the discrete set of belief points β;
- A is the set of actions of the original BMDP;
- $\tau : B \times A \mapsto \Pi(B)$ is the transition function between belief points of the discrete set B;
- $\rho(\beta, a)$ is the reward function on belief points on B, with $\rho(\beta, a) = r(b, a)$.

We will use the notation

$$\beta(s) = Pr(x_1)Pr(x_2)...Pr(x_n)$$

in the same way of $b(s)$.

Once the discretization process is performed, we obtain a DBMDP model of the Cooperation POMDP. We can then solve it using a classic MDP resolution technique, such as Value Iteration or Policy Iteration (Algorithms 1 and 2).

4.2.2 Reintroducing observations: the Belief Shift function

The belief transition function $t(b, a, b')$ and the following discretization of the belief state can usually be computed during execution time. However, this is not possible if we want to implement the POMDP policy as a Finite State Machine. In such a case, the system needs to precompute all belief transitions conditioned on the acquired observation. In order to do so, we need to unbind the observations from the belief point computation.

We suggest the introduction of a *belief shift function* σ, defined as $\sigma : B \times A \times \Omega \mapsto B$, and depicted in Figure 4.2.

The belief shift $\sigma(\beta, a, o, \beta')$ is a deterministic transition function between belief points, conditioned on the observations. It is computed in the following way: given the input belief point β, which corresponds to a belief state b of the original POMDP, and a given action a, we compute the updated belief state $\tilde{b}^a_o$ for each $o \in \Omega$, using the standard belief update (Equation 3.2) reported below:

$$\tilde{b}^a_o(s') = \frac{\sum_s O(o|s,a,s')T(s'|s,a)b(s)}{\sum_{s'}\sum_s O(o|s,a,s')T(s'|s,a)b(s)} \quad \forall s' \in S$$

$\tilde{b}^a_o$ is the resulting belief when executing action a in belief state b and observing o. It represents a probability distribution over S: hence, a discretization process is required to translate $\tilde{b}^a_o$ into a belief point β'. That is:

$$\beta' = \arg\min_{\beta} \quad dist(\tilde{b}^a_o, \beta) \tag{4.2}$$

where $dist(\tilde{b}^a_o, \beta)$ is the distance between $\tilde{b}^a_o$ and β, $\forall \beta \in B$. As a distance between probability distributions, $dist(\tilde{b}^a_o, \beta)$ can be computed in several ways, such as Euclidean distance, Hellinger distance, or using the Kullback Leibler divergence.

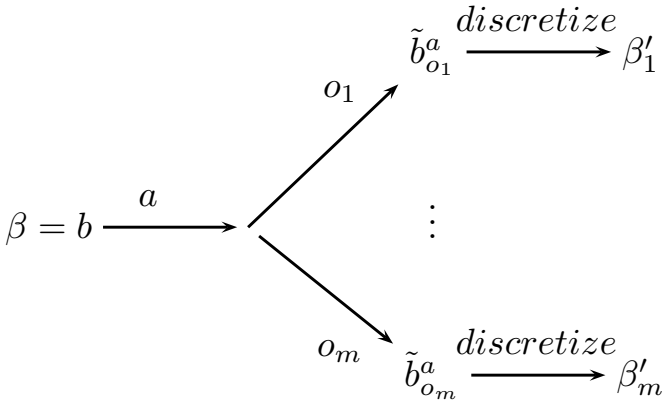


FIGURE 4.2: The belief shift function

Therefore, we have obtained a function $\sigma(\beta, a, o, \beta')$, which describes the state transitions between belief points conditioned on the received observation. To obtain the traditional BMDP transition function τ, we simply sum over the observations:

$$\tau(\beta, a, \beta') = \sum_{o \in \Omega} \sigma(\beta, a, o, \beta') P(o|a, \beta, \beta') \tag{4.3}$$

To summarize the described process, we highlight how the policies are defined and executed by the different models.

In a POMDP, a policy tree can be implemented as a FSM, but only with finite-horizon policies.
In a Belief-MDP, the policy associates an action to each belief point. The observation function is included in the belief transition function τ. When the system executes an action, the belief is updated according to the sensed observation. Such a belief update cannot be performed within a FSM during execution, since the domain of beliefs is continuous and the FSM would require an infinite number of states.
Therefore, we use the belief shift function to recover the POMDP policy from a DBMDP policy and thus to make the DBMDP policy compatible with FSMs.

4.2.3 Translating the DBMDP policy into a FSM

Once the transition model is computed, the Discretized BMDP can be solved using classic MDP techniques such as Value Iteration or Policy Iteration. Solving the BMDP modeled with the τ transition function results in a policy $\tilde{\pi}$, which associates an action a to each belief point β. Solving the DBMDP using the belief shift function instead results in a policy π^{σ} which associates an action a to a list of possible belief points $(\beta_1, ...\beta_m)$, with $m = |\Omega|$.

To convert $\tilde{\pi}$ into an executable FSM, the effect of observations must be stated explicitly. It is possible to perform a conversion of the $\tilde{\pi}$ into a π^{σ} in order to make it compatible with a FSM.

We represent $\tilde{\pi}$ in the following way:

Definition *(DBMDP policy)* 4.1. A Discrete-BMDP policy $\tilde{\pi}$ is a function $B \rightarrow A$. It is represented as a set of $|B|$ belief-action pairs $< \beta_i, a_i >$ such that $\tilde{\pi}(\beta_i) = a_i$.

π^{σ} is the POMDP policy, which is defined in the same way as the DBMDP policy, but with the addition of a set of multiple outcomes:

Definition *(POMDP policy)* 4.1. A POMDP policy π^σ is a function that associates to each belief $\beta \in B$ an action a and a set of belief outcomes $SS^{\tilde{\pi}}$: $\pi^\sigma(\beta_i) =< a_i, SS_i^{\tilde{\pi}}$. $SS_i^{\tilde{\pi}}$ is the set of outcome beliefs resulting when applying the belief shift function to an input belief β_i for a given DBMDP policy $\tilde{\pi}$:

$SS_i^{\tilde{\pi}} = \{\beta'_i = \sigma(\beta_i, \tilde{\pi}(\beta_i), o)\}$ and $|SS_i^{\tilde{\pi}}| = |\Omega|$.

Data: policy $\tilde{\pi}$
Result: policy π^σ
forall *input state* $\beta_i \in B$ **do**
  action $a_i \leftarrow \tilde{\pi}(\beta_i)$;
  $SS_i^{\tilde{\pi}} \leftarrow \emptyset$;
  forall *observation* $o \in \Omega$ **do**
    $\beta'_i \leftarrow \sigma(\beta_i, o, a_i)$;
    $SS_i^{\tilde{\pi}} \leftarrow SS_i^{\tilde{\pi}} \cup \beta'_i)$;
  end
end
return $\{< \beta_i, a_i, SS_i^{\tilde{\pi}} >\}$

Algorithm 3: Translating a DBMDP policy into a POMDP policy

Algorithm 3 selects for each input belief β_i the action given by the policy $\tilde{\pi}$, then computes the belief shift for each observation.

The resulting policy π^σ therefore provides for each belief point β_i an action a and an outcome belief β'_i for each observation. The policy is in general sub-optimal. While $\tilde{\pi}$ may constitute an optimal policy for the Discrete-BMDP, π^σ does not usually constitute an optimal policy for the original POMDP, since the discretization of the Belief-MDP is essentially an approximation process.

4.3 Chapter Conclusions

In this Chapter, we have presented a novel approach for ensuring the human's cooperation within a joint activity. The approach uses a hierarchical structure that abstracts the domain's state-space to reduce the planning complexity. The main contribution of the approach is the decomposition of the joint activity into two separate aspects, the Task itself and the human-robot Cooperation aspect. Each aspect is handled by its own module in the framework. We additionally described a method for solving the Cooperation module. We model the Cooperation POMDP as a Belief-MDP and the discretize the belief space. We adopt a factored discretization method to customize and reduce the

resulting set of belief points. We then solve the Discrete-BMDP model with Value Iteration or Policy Iteration and obtain an infinite-horizon state-action policy $\tilde{\pi}$. To make the policy executable, we have defined a belief shift function $\sigma(\beta, a, o, \beta')$ that returns the output belief point β' resulting from performing action a in belief β and receiving observation o. We then use this function to translate the state-action policy $\tilde{\pi}$ into a infinite horizon belief-action-observation POMDP policy π^{σ} .

This approach presents several advantages.

- Flexibility: layers and modules may be modeled and implemented separately.
- Offline planning: there is no need to perform re-planning during mission execution for the Cooperation module
- Cross-application: the same Cooperation plan can be re-used for different instances of the application. For example, if a guide robot is deployed in a different environment, the Cooperation plan does not change and only the Task aspect must be updated. The presented approach can easily adapt to different HRI applications as well.
- Infinite horizon policy: there is no bound on how many time steps the robot is able to plan for.

Chapter 5

The Escort Task application

5.1 The Escort Task scenario

We will now describe our implementation of the proposed framework in an application scenario. This scenario is the testbed application of the European project COACHES. The scenario consists in the deployment of a mobile service robot in a shopping mall. When a customer asks assistance for reaching a point of interest (POI), the robot offers to physically guide and escort him along the way. While the project covers several domains and adresseses several challenges, the contribution of the thesis focuses on the Escort task, where the robot needs to guide the user towards the POI.

5.1.1 Overview of the robots

We will now describe the two robots that were used in the course of the thesis as implementation platform. The two robots, named Cadomus and Romus, have an identical hardware and software setup. The robot's architecture is comprised of several components and is the result of the collaboration between the University of Caen, University La Sapienza of Rome, the Sabienci University of Istanbul and Vrije University of Bruxelles.

The Cadomus robot (and his brother Romus), shown in Figure 5.1, is built on a segway RMP-210 base [1]. They are both 147 cm high, 65 cm wide from left to right wheel, and 78 cm long at the base. They provide several sensors:

[1] Cadomus and Romus were designed and produced by Algorithmica

- Front and Rear laser scanners
- One RGB camera on the rear.
- Two RGBD cameras, on the front and on the top of the robot
- A touchscreen interface
- Audio speakers
- A microphone

FIGURE 5.1: The Cadomus robot at the Rives de l'Orne shopping mall

5.1.2 The COACHES architecture

Figure 5.2 shows the software architecture of the robots. It describes the information flow among the different modules of the architecture, from sensors input to actuator execution. Most of the architecture has been implemented using the Robot Operating System (ROS) [2] middleware, which is a standard and widely popular middleware for interfacing robot sensors and cross-platform control modules. We do not detail the ROS

[2] `http://www.ros.org/`

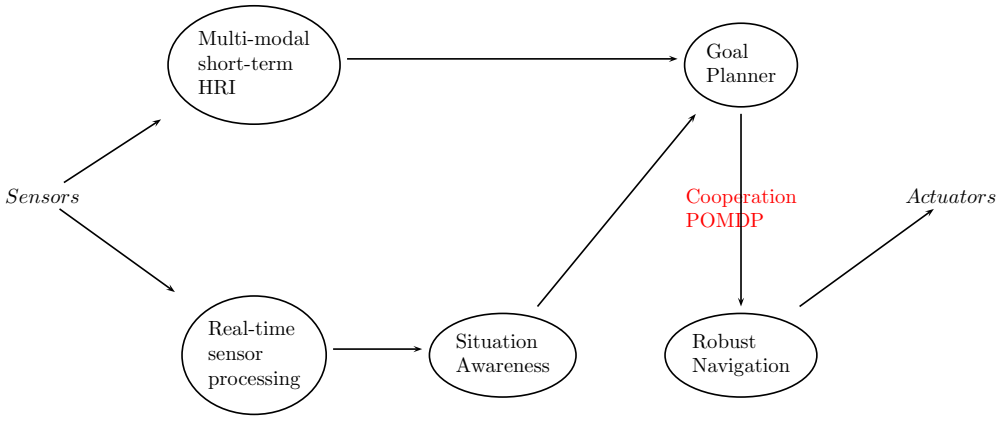


FIGURE 5.2: The software architecture of the robot

implementation of the architecture. We only briefly describe each module to present its role in the overall system, highlighting their impact on the thesis work.

5.1.2.1 Multi-modal HRI

This module performs multi-modal interaction with the user. Its main interface is the touchscreen tablet. A custom Python Graphical User Interface (GUI) allows the user to select the robot's services, and specifically the goal location where he wishes to go, thus starting the escort task. The GUI also allows the robot to greet people, provide useful information and advertisement about the mall and its shops, and display any message required during its tasks. The GUI is also available as a web-based service to allow users to select their destination using their smartphones.

The short-term HRI module also uses a speech synthetizer to output any message it generates as spoken dialogue. The module can perform speech recognition as well and understand simple vocal inputs. This capability however has not been fully exploited in the testbed scenarios and is only considered as an optional feature.

The module can personalize the dialogue to the user, notably for setting the language of displayed and spoken messages. More generally, it is able to store several parameters that define a user profile, such as language, gender or age (children or adult), and provide dialogue specific for the profile. More details about the personalized interaction can be found in [101].

5.1.2.2 Video and sensor processing

This module processes the information provided by the 2D and 3D cameras and by the laser scanners.

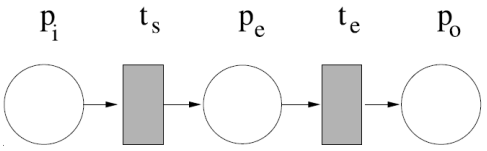
It uses Compressive Tracking technique on RGB video data to detect and track people within the robot's field of view. It also performs people tracking with the laser scans by identifying feet positions. These data are fused together for a more robust people tracker. Because they have a wider field of view, laser scans are particularly effective when the tracked person is within the blind spot of the cameras (the left and right of the robot). The module is also able to provide the person's orientation and distance with respect to the robot, which is especially useful for providing the proxemic distance of the user and estimating his level of attention during the escort task.

5.1.2.3 Goal planner

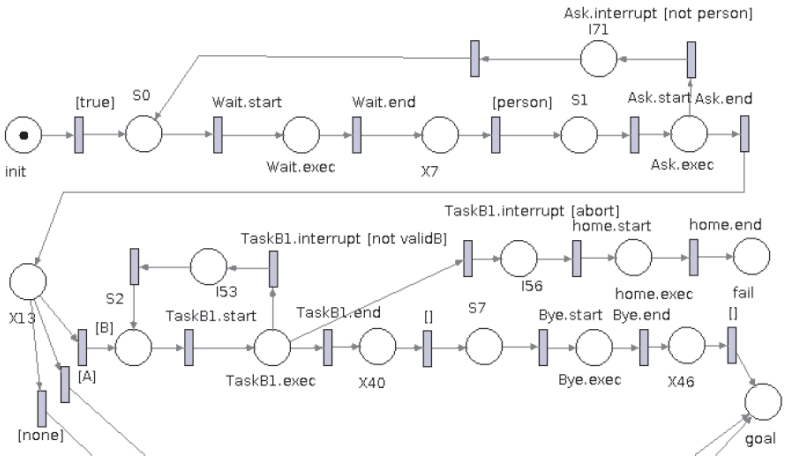
The role of this module is to generate and schedule tasks and goals. When the robot detects a new customer entering the mall, it may show a proactive behavior and start an assistance task. When the user selects a destination on the GUI, the goal planner starts an escort task with the selected POI as destination. A knowledge base translates the semantic POI into a specific location in the mall's map that the navigation module may reach. This module is especially important in the case of multi-robot task scheduling. The high-level task planning is performed using MDPs and Progressive Reasoning Units (PRUs), a formalism introduced in [102] and applied to the COACHES project in [103]. For the scope of the thesis, the contribution of the Goal Planner is to initiate the Escort plan, including both Task and Cooperation sub-systems, and to provide a destination to reach.

5.1.2.4 Petri Net Plans

The robot's architecture uses Petri Net Plans to execute the high-level plans generated by the Goal Planner. Petri Net Plans (PNPs), developed by Ziparo et al. [104], are robot plans represented as Petri Nets. They are an extension to Finite State Machines and as such they can implement the policy generated by the Cooperation POMDP.



(a) An elementary PNP action



(b) A sample PNP task with Execution Rules included

FIGURE 5.3: Examples of Petri Net Plans. (Images from [104])

A Petri Net is a directed weighted bipartite graph. Its nodes can be of two types: *places* and *transitions*. A *place* may or may not contain tokens, whose number on a given place describes the current state of the system. *Transitions* represent events and are always connected to a set of input places and a set of output places. When the associated event occurs, the transition fires, and moves tokens from the input places to the output places. *Edges* connect two places by passing trough transitions. Edge *weights* define the minimum number of tokens required in an input place to enable the associated transition to fire.

Exploiting the expressiveness of Petri Nets, PNPs are able to model complex robot behaviors. PNPs are able to model robot actions with a high level of detail. Specifically, because real-world execution of robot actions is not instantaneous, each action is composed of at least three places and two transitions, which represent different phases of the action execution: an *initial* place, an *action starting* transition, an *execution* place, an *action termination* transition and a *termination* place (rFigure 5.3(a)). This elementary structure can be expanded with additional places and transitions to implement more complex behavior, such as observation sensing, action interrupts, loops, concurrent execution, and so on.

The main advantage of using PNPs is the robustness of plan execution. Following [103], Execution Rules are included in the PNP to add interrupt transitions that may occur during execution (Figure 5.3(b)). These rules define which kind of recovery behavior the robot must follow whenever the action is interrupted during execution: either restart or skip the action, restart the whole plan, or consider the plan terminated as a failure. Any variable and parameter related solely to the execution of the task is therefore delegated to the Execution Rules and thus excluded from the planning phase. Implementing MDPs and POMDPs as PNPs therefore allows to separate execution concerns from the planning model and thus reduce computational complexity.

5.2 The Escort POMDP

We will now describe how the hierarchical framework introduced in Chapter 4 was implemented for the Escort Mission. The Escort is a collaborative task since both human and robot need to reach the goal destination together, ensuring all along the task that the user doesn't get lost. Therefore, we consider this scenario to be a suitable application of the proposed framework. Specifically, we focus on how the Cooperation layer fits in the Escort scenario and we describe in detail the POMDP model of the Cooperation module.

5.2.1 Overview

The implemented framework follows the one described in Section 4.1.1. The Escort Mission consists of a Navigation task and a Cooperation problem. While the Navigation concerns itself with reaching the destination, the Cooperation aspect tries to ensure that the user is intent on following the robot. For implementation purposes only, in our example application we merged the Mission Status layer with the Cooperation sub-system, since they share the same state-space. The Cooperation module therefore performs three roles:

- It acts as an observer for the current state of the joint intention between human and robot.
- It provides a policy for re-establishing cooperation when missing.

- It decides whether activating the Navigation module or executing the Cooperation's policy action.

The Primitive layer for the Escort Mission consists at its core of a navigation domain. We formalize it as a simple grid world domain with discrete coordinates. The main variables that define the states at the Primitive level are the robot's coordinates and orientation R_x, R_y, R_θ, the human's coordinates H_x, H_y, and level of attention Att, and the goal's coordinates G_x, G_y.

As already mentioned in Section 4.1.1, we do not detail the implementation of the Navigation part of the Escort Mission. The COACHES architecture already contains a Navigation component that can be used during the escort. Instead, we focus on detailing the POMDP model for the Cooperation aspect.

5.2.2 State-space

The state-space of the Cooperation POMDP for the escort mission is built in a factored way using three main independent variables: the *attention level*, the *proxemic interaction distance*, and the *relative position* (Figure 5.4).

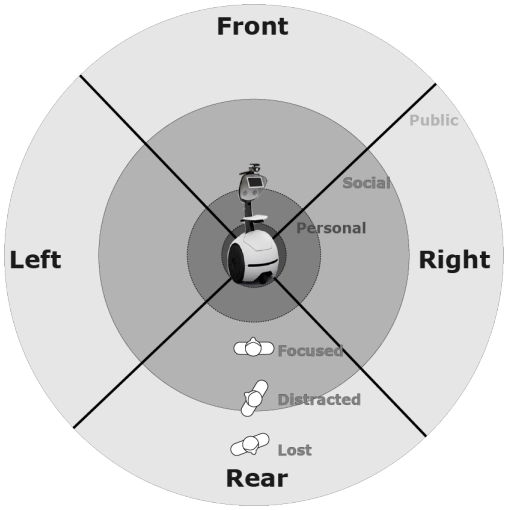


FIGURE 5.4: The state-space of the Cooperation POMDP

Attention Level

The Attention Level, *Att* represents the mental state of the user and his current level of engagement to the shared task. It can be estimated by detecting and tracking the

human's head and gaze orientation, using video processing techniques to determine if the human is concentrated on following the robot. The Attention level consists of the following values:

- *Focused*: the human's gaze is focused on the robot.
- *Distracted*: the human is slightly distracted. He may be looking at the shops nearby, or at his phone, or similar cases.
- *Lost*: the human is completely neglecting the robot. He may be turning back, or concentrating on some activity other than the joint task.

Proxemic Interaction Distance

Proxemic Interaction Distance, *Dist*, is the relative distance of the human w.r.t. the robot. We use the studies on Proxemics described in Section 2.2.1 to define a set of interaction distances: *Intimate*, *Personal*, *Social* and *Public*. Their main use is to help in better understanding the human's behavior: if the human stays too far from the robot, it may mean that he needs help or that he is going away, while if he wanted to follow the robot he would probably stay within the *Personal* space or *Social* space, instead of *Intimate*. Proxemic spaces were introduced in the study of human-human interactions, and some adjustments are required to adapt them to human-robot interactions. For increased safety, we have slightly changed the distance values in the following way:

- *Intimate*: between 0 and 60 cm.
- *Personal*: between 60 and 160 cm.
- *Social*: between 1.6 and 3.7 m.
- *Public*: between 3.7 and 7.6 m. (and beyond)

Relative Position

In addition to the distance, also the Relative Position *Pos* of the human w.r.t. the robot is taken into account. The human may be situated on the *Front*, *Rear*, *Left* or *Right* side of the robot. Intuitively, a person following the robot would be mostly on the *Rear*

side of the robot.

Remark that we assume the independence of the variables. Interaction distances are defined as constant regardless of the human's Relative Position. The state-space of the Cooperation POMDP therefore is built as $S = Dom(Att) \times Dom(Dist) \times Dom(Pos)$ and consists of 48 states.

Within the generated state-space, we then define the Cooperative States set. This set contains those states where joint intention is ensured and the human is currently engaged in the shared task. A *Lost* attention level trivially means that the human is not cooperating with the robot, and thus negates the joint intention. Similarly, a *Public* value of the Interaction Distance indicates a distance that is too far for a good cooperation. Therefore, as long as the user is *Focused* on the robot and not at a *Public* distance, he is considered to be cooperative. Also, states with a *Distract* attention value, a *Personal* or *Social* distance and a *Front* or *Rear* position belong to the CS set as well.

5.2.3 Actions and Rewards

The actions that the Cooperation POMDP may plan for are the following:

- *pause*: the robot stays idle and does not move
- *move forward*: the robot moves straight forward along its current direction. Since this is an abstract macro-action, actual execution of the movement, its speed setting and obstacle avoidance process is handled by a low-level ROS module.
- *turn left* and *turn right*: the robot turns on the spot. Similarly to move forward, details of low-level execution are not detailed at the Cooperation level of abstraction.
- *draw attention*: the robot speaks and asks the user to follow it. The user's Attention Level becomes *Focused*.
- *navigate*: this action activates the Navigation module, which moves the robot along the planned path towards the destination.

In order to switch between the Navigation and Cooperation sub-system, the *navigate* action is introduced in the Cooperation POMDP. *Navigate* is a virtual action, a macro-action corresponding to the navigation policy. It is used to implement the Mission Status level switching conditions described in 4.1.2. While computing the optimal policy, the POMDP model will hence be able to determine the best moments for changing the sub-system. During execution, the effects of the virtual action depend on which sub-system is currently active.

The reward function $R(s, a)$ assigns a fixed cost for all actions, with a smaller cost for the *pause* action. It assigns a great reward +50 whenever a *navigate* action is performed in a state belonging to the CS set. In such way, the optimal plan will proceed with the navigation task towards the destination when cooperation is ensured, and re-establish the joint intention otherwise.

5.2.4 Transition Function

In order to build the transition function for the Cooperation POMDP, we need to define the behavioral model for the human. In order to represent the unpredictability of the human behavior and his low level of commitment, we do not consider him to behave according to a fixed policy. Instead, we model his behavior in the same way as an autonomous environmental variable, which evolves in a probabilistic way and can only be controlled partially by the robot. Since the POMDP is built in a factored way, we can define separately the attention model and the movement model of the person.

5.2.4.1 The human movement model

The position of the person changes with a given probability that depends on his current level of attention and distance from the robot. We have defined three actions that the person may perform in the Primitive layer:

- *stay*: the person does not move.
- *move to robot*: the person moves one step towards the robot.
- *deviate*: the person moves one step in a random direction.

Given the coordinates of the human H_{xy} and robot $R_{xt\theta}$, we give a probability for each human movement:

$$P_{stay} = Pr(\text{stay}|Att, H_{xy}, R_{xt\theta})$$

$$P_{move} = Pr(\text{move}|Att, H_{xy}, R_{xt\theta})$$

and

$$P_{deviate} = Pr(\text{deviate}|Att, H_{xy}, R_{xt\theta})$$

The three human movements are mutually exclusive.

We use these probabilities to define the Primitive level transition function T^{bot} for the human position variable:

$$Pr(H_{xy}^{t+1}|Att^t, H_{xy}^t, R_{xt\theta}^t) = f(P_{stay}, P_{move}, P_{deviate})$$

Remark that the human's movements do not depend on the robot's actions.

For our Escort POMDP, however, we need to define the human's movement within the Cooperation level transition function T^{co}, defined over the abstract variables *Attention level*, *Proxemic Interaction Distance* and *Relative Position*:

$$Pr(Dist^{t+1}|Att^t, Dist^t, Pos^t, a)$$

$$Pr(Pos^{t+1}|Att^t, Dist^t, Pos^t, a)$$

Because we are now dealing with relationships between human and robot, the Cooperation level transitions depend on the robot's actions.

In order to do so, we need to abstract the state-space $S^{bot} \mapsto S^{co}$. We abstract the grid world of the Primitive layer into zones Z, which correspond to the proxemic distances (Intimate, Personal, Social and Public). By checking the distance between the human's coordinates H_{xy} and the robot's position R_{xy} in each state of the Primitive level, we define N_Z as the number of states belonging to each Zone. In addition, each zone is divided in three regions: a border region $F_+(Z)$ with the next zone (closer to the robot), one $F_-(Z)$ with the previous zone (further from the robot), and a middle region $F_0(Z)$. Each border is defined as the region of a zone where a single discrete step is sufficient to go into another adjacent zone. Trivially, the Intimate zone does not have a border with

a successive zone, while the Public one does not have a border with a previous zone. By counting the number of states $N_{F(Z)}$ belonging to a particular zone region, we can compute the conditioned probability of the human position H_{xy} to be in said region as

$$Pr(H_{xy} \in F(Z) | H_{xy} \in Z) = \frac{N_{F(Z)}}{N_Z}$$

with $F(Z) \in \{F_+(Z), F_0(Z), F_-(Z)\}$.

We can thus compute the probability that the human's movement reduces or increases the interaction distance. For instance, if the human's position belongs to the next border region $F_+(Z)$ of the *Social* zone and his movement is *move to robot*, then the Proxemic Interaction Distance will become *Personal*.

We can therefore compute the probability that the human reduces the interaction distance, that is, moves from one zone to the successive one, as:

$$P^H_{Zsucc} = Pr(H_{xy} \in F_+(Z) | H_{xy} \in Z) \, (P_{move} + \phi P_{deviate})$$

where ϕ is the probability to approach the robot given the *deviate* random movement. We can define in a similar way the probability for the human to increase the interaction distance, P^H_{Zprev} and to stay within the same proxemic zone P^H_{Zsame}. We also need to account for the robot's movements. We define in a similar way $P^R_{towards}$, P^R_{away} and P^R_{stay} the probabilities that the robot is approaching, moving away or staying at the same distance of the human, respectively. These probabilities depend on the robot's direction R_θ, on its current action a and on the agents' coordinates H_{xy}, R_{xy}.

The complete probability to move from one Proxemic Interaction Distance to the successive is therefore:

$$P_{succ} = P^H_{Zsucc} P^R_{towards} + P^H_{Zsucc} P^R_{stay} + P^H_{Zsame} P^R_{towards}$$

$$P_{prev} = P^H_{Zprev} P^R_{away} + P^H_{Zprev} P^R_{stay} + P^H_{Zsame} P^R_{away}$$

$$P_{same} = 1 - P_{succ} - P_{prev}$$

We have thus obtained a transition function for the Proxemic Interaction Distance variable $Pr(Dist^{t+1} | Att^t, Dist^t, a)$. The dependence from the attention level Att results

FIGURE 5.5: Attention Level transitions.

from the human's movements *stay*, *move* and *deviate*, the dependence from the interaction distance *Dist* results from the abstraction in zones $H_{xy} \in Z$, and the dependence on the action a results from the robot's contributions $P^R_{towards}$, P^R_{away} and P^R_{stay}.

A similar approach is adopted for the Relative Position variable as well. We divide the primitive level's state-space into zones *Front*, *Left*, *Right* and *Rear*, then we compute the probability to change zone given the human's movements *stay*, *move* and *deviate*, the probability of being on a frontier between zones, and the robot's actions (especially *turn left* and *turn right*).

5.2.4.2 The human attention model

The Attention Level variable does not require any abstraction process from the Primitive layer to the Cooperation layer. It evolves autonomously according to the model in Figure 5.5 regardless of human-robot distance and position.

Robot actions do not influence the Attention Level variable, with the exception of the *draw attention* action. This action changes the human's attention to *Focused* with a given probability $Pr(\text{changeToFocus}|Att, Dist, drawAttention)$ that varies according the Attention Level and Interaction Distance.

5.2.5 Observations

The cameras and laser scanners mounted on the robots can provide the observations required for the Cooperation POMDP. Specifically, they can provide the following information:

- Detect the person. The video and sensor processing module fuses data from the cameras and the laser scanners to detect and track the position of persons around

the robot. The sensors have a limited field of view, therefore there are blind spots on the robot's sides where the user cannot be detected.

- Compute the distance. If a person is detected, then his distance from the robot can be obtained. The range of the detection process is limited below 3.7m, so persons can not be detected at the *Public* interaction distance.
- Detect the person's face. If a person is detected by the cameras, we can estimate his level of attention by checking whether he is oriented towards the robot or not.

The position, the interaction distance and the attention level of the user are not independently observable features, since it is not possible to observe neither the distance nor the attention level if the user is not detected. Nevertheless, we model the POMDP observations as the combinations of the following information:

$$Position\_obs = \{oFront, oRear, oNoDetection\}$$

$$Distance\_obs = \{oIntimate, oPersonal, oSocial, oNoDistance\}$$

$$Attention\_obs = \{oLook, oNoLook\}$$

We use these observation features to define a set of observations Ω, ensuring that it does not contain impossible observations. Since the robot cannot extract the attention and distance information if the person is not detected by the front or rear cameras, the observation set Ω is built in such way that the resulting observation can only have a $oNoLook$ and $oNoDistance$ value if $Position\_obs = oNoPosition$. Once the observations are defined, we can build an observation function $O(o|a, s')$.

A later improvement led to the introduction of additional information. Because we have partitioned the Mission into two independent modules, the Cooperation module has no knowledge about the goal's location, and as a consequence no knowledge about the actions performed by the Navigation module either. This may affect the update of beliefs. Compare the following cases: first, the robot is situated between the user in front of it and the destination behind it. As soon as the navigation starts, the robot turns to face the goal and approaches it. In the second case, the user is behind the robot and the destination straight ahead. In both cases, the action executed is the same, *navigate*, but the outcome state s' is different. There is no information about the destination or

the starting state s. As such, the probabilities to observe the position or distance of the user when the *navigate* action is performed may be incorrect.

Including information about the goal's location would contradict the principle of separating the Navigation and Cooperation aspects of the Escort Mission, and would require to recompute the Cooperation plan whenever a new destination is selected. Instead, we include the starting state in the definition of the observation function $O(o|s, a, s')$. We also introduce as an additional observation the *current action performed*. This solution allows us to observe the actions performed while the Navigation module is active, and to update beliefs more accurately. Therefore, we include in the Ω set an additional and independent observation feature

$$Action\_obs = \{oTurningLeft, oTurningRight, oMovingForward, oWaiting\}$$

5.2.6 The Discrete Belief-MDP

Following the approach described in Section 4.2.1 we build a Discrete-BMDP trough a discretization process of the belief space. We use a discretization factor of $k = 4$, meaning that belief probabilities may only belong to the following set of values: $\{0, 0.25, 0.5, 0.75, 1\}$ with the addition of value $\frac{1}{|Dom(x)|}$, introduced to allow for an uniform probability distribution.

A discrete set of probability distributions is generated for each variable separately. This factored approach allows us to reduce the number of belief points generated. Additionally, we make the assumption that the system may only have an uncertainty about adjacent values. This is an intuitive assumption for those variables whose values have a spatial meaning and can be ordered sequentially. For instance, the domain of the Proxemic Interaction Distance is ordered in the following way:

$$[Intimate, Personal, Social, Public]$$

We only include distributions where non-zero probabilities are adjacent to each other. For instance, the distribution $[0.5, 0, 0, 0.5]$, where the user may either be at an Intimate distance or at a Public distance, is not included. This assumption, however, cannot be made for the Relative Position variable.

The complete discrete belief space is then generated as the product of those probability distributions. For example, the belief point corresponding to an uniform distribution over the state-space can be represented as:

$$\begin{aligned}\beta_0 &= (U(Dom(Att)), U(Dom(Dist)), U(Dom(Pos))) \\ &= [0.33, 0.33, 0.33, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25] \quad (5.1)\end{aligned}$$

where $U(A)$ denotes an uniform distribution over the elements of A.

This representation corresponds to an uniform probability over S:

$$b(s) = \beta_0(s) \approx 0.0208 \quad \forall s \in S$$

Using this approach, given the POMDP model with $|S| = N = 48$ states and $k = 4$ as the discretization factor, we have generated $|B| = 4480$ belief points. Following Equation 4.2.1, an uniform discretization of the belief space over S would have generated 249900 belief states.

$$\binom{48 + 4 - 1}{4} = \frac{51!}{47!4!} = 249900$$

Not only our approach reduces significantly the complexity of the model, but it also allows to define a non-homogenous distribution of belief points, as well as including the uniform distribution β_0 (which would have otherwise required a discretization factor of $k = 48$). These belief points constitute the state-space of the Discrete-BMDP. We build the transition function using Equation 4.3:

$$\tau(\beta, a, \beta') = \sum_{o \in \Omega} \sigma(\beta, a, o, \beta') P(o|a, \beta, \beta')$$

The reward function is computed as $\rho(\beta, a) = \sum_{s \in S} R(s, a)\beta(s)$.

The DBMDP is solved using Value Iteration. The resulting policy is then translated into a POMDP policy π^σ following the process described in Algorithm 3.

5.2.7 Generating the PNP

Once the policy is translated into π^σ, it is possible to generate a Finite State Machine. The COACHES architecture uses Petri Net Plans to execute plans. Therefore we need to translate the generate policy π^σ into an executable PNP. The process is described in Algorithm 4 and it follows the approach by Iocchi et al. [103], with the remark that the POMDP observations are used as execution conditions associated to the corresponding outcome states. Each input belief point β has a branching factor of $|\Omega|$. By observing o, the PNP executor can determine the successor belief point β'. In the following algorithm, s denotes a state of the generated PNP, $SS_i^{\tilde{\pi}}$ is the set of outcome beliefs from belief β_i, and SS_i is the set of PNP outcome states from state s_i, represented as state-observation pairs $< o, s >$.

Data: $\pi^\sigma =< \beta_0, G, \{< \beta_i, a_i, SS_i^{\tilde{\pi}} >\}$
Result: PNP implementing π^σ
$s_0 \leftarrow \beta_0$;
push(Q,s_0) ;
$p \leftarrow$ empty_PNP ;
$V \leftarrow \emptyset$;
while $Q \neq \emptyset$ **do**
  $SS_i \leftarrow \emptyset$;
  $\beta \leftarrow$ pop(Q) ;
  select $< \beta_i, a_i, SS_i^{\tilde{\pi}} > \in \pi^\sigma$;
  $s_i \leftarrow \beta_i$;
  forall $o \in \Omega$ **do**
    $s' \leftarrow \sigma(\beta_i, o, a_i)$;
    $SS_i \leftarrow SS \cup < o, s' >$;
  end
  $p \leftarrow PNP\_add(p, < s_i, a_i, SS_i >)$;
  forall $s' \in SS_i$ **do**
    if $s' \notin V$ **then**
      $V \leftarrow V \cup \{s'\}$;
      push(Q, s') ;
  end
end
return p

Algorithm 4: Translating a POMDP policy into a PNP

Because the Cooperation sub-system and the Mission Status layer have been implemented together, the generated PNP also includes the switching instructions that allow the robot's architecture to execute the Navigation module's actions or the Cooperation policy. This is performed by augmenting the PNP with a boolean variable, $NavStatus = \{NavOFF, NavON\}$, that describes whether the Navigation sub-system

is currently active during execution or not. Remark that such augmentation is performed after the policy computation and thus does not affect the computational cost of the planning phase. Also remark that there is no partial observability on the boolean variable.

The whole plan generation process for the Cooperation DBMDP is performed offline. The result is a Petri Net Plan file that can be loaded into the robot's architecture.

5.3 Execution

We describe now how the whole proposed framework is actually executed on the robots. The execution process of the framework is shown in Figure 5.6.

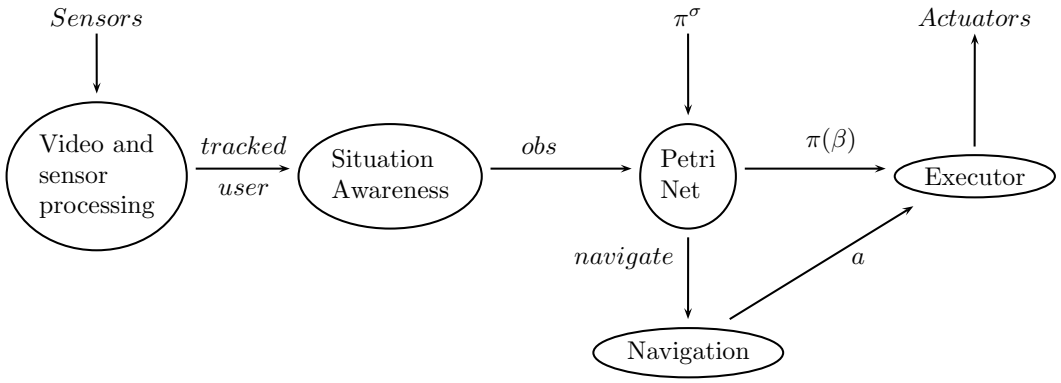


FIGURE 5.6: Execution architecture

The PNP Manager loads high-level plans and executes them. Like any Finite State Machine, it handles state transitions conditioned on the observations received. The actual execution of actions is performed at a lower level by the Executor. The Navigation module receives a goal destination from the Goal Planner, described in Section 5.1.2, and performs path-planning to reach it. Dynamic obstacle avoidance is managed by the Executor.

In order to provide the observations required by the Cooperation plan, a Situation Assessment module has been implemented within the ROS architecture of the robot. It acts as an interface between the Video and Sensor Processing module and the PNP manager. The sensor processing module detects and tracks the user, providing his coordinates in the robot's reference frame. The Situation Assessment module translates the

coordinates into a *Position_obs* observation. It also computes the distance and translates it into a *Distance_obs* value. Whenever the video processing module detects the user's face, the Situation Assessment translates this information as a *oLook* observation. Lastly, it also provides the *Action_obs* information by retrieving the action currently executed. All observed information is then put together in a string of characters and sent as observation to the PNP executor. These observations act as conditions in the PNP and trigger the state transitions.

For simple tasks, when the robot needs to reach a destination it keeps following the path provided by the Navigation module. When an Escort task starts, however, the Cooperation plan is loaded as a Petri Net Plan, π^σ. The plan is a file generated offline: only the Navigation module's path-planning process computes a new plan whenever a different destination is selected. The Cooperation plan acts as an interruption in the normal navigation routine. The plan first observes the current state of human-robot cooperation and provides the optimal action $\pi^\sigma(\beta)$ for the current belief β of the cooperation state. Most of the actions described in Section 5.2.3 are executed normally; the *navigate* action, however, is a virtual action. Whenever the PNP manager finds a *navigate* action, it starts or proceeds with the Navigation module, which plans the path towards the destination and the actions to reach it. As long as the Escort Mission is running, the Cooperation plan is always active. Even when the robot is executing the Navigation module's actions, the Cooperation PNP keeps on updating its belief on the state of the joint intention, so that it may execute the appropriate action when $\pi^\sigma(\beta)$ is no longer a *navigate* action.

5.4 Group Escort

The proposed approach for the Escort Mission can be extended to groups of people. Three conceptual models can be defined when guiding groups of people:

- **Leader model:** one of the members of the group acts as a leader. The robot can therefore track and guide the leader as in the single-user escort scenario.
- **Single-entity model:** the group is treated as a single entity. The Situation Assessment module computes the average Attention Level of all members of the

group, as well as the position and interaction distance with the centroid of the group. The Cooperation module then tries to ensure the cooperation of the group as a whole, as in the single-user case, using the average values of the group.

- **Multiple tracked users model:** this is the most complex case, since it treats the cooperation of each group member separately. The hierarchical structure of the framework can adapt to such scenario: the Mission Status layer would estimate the cooperation of the group as a whole, treating it as a single entity. Whenever Joint Intention would not be ensured, it would need to understand which members of the group are not cooperating and decide with whom to interact with. Once the Mission Status layer has decided to re-establish cooperation with a specific person, however, the single-user cooperation policy can be re-used for that person.

5.5 Chapter Conclusions

In this Chapter we have shown how the novel framework can be implemented on a real application. The thesis has been carried out within the scope of the COACHES project and implemented on the robots Cadomus and Romus. The hardware and software architecture of the robots has been described to provide a context for the implementation of the proposed framework.

We have presented the Escort scenario, where the robot has to guide the user to a desired destination. A Cooperation POMDP has been modeled to ensure that the user is engaged in the Escort task and keeps following the guide robot. We have detailed the state-space of the model, built from the *attention level*, the *proxemic interaction distance*, and the *relative position* state variables. We have described the actions, rewards, observations and transition function of the POMDP model. Each has interesting features that are specific to the implementation to the Escort task:

- For the transition function, a behavioral model for the human was defined. The probabilites of the human's actions were defined on a low-level state-space, and then an abstraction process was performed to compute the probabilities at the abstract Cooperation level. This abstraction process converts information about low-level position coordinates into information about relationships between human

and robot. This approach differentiates itself from most hierarchical POMDP frameworks for robot navigation: higher-level states do not represent the same kind of information as lower-level states. For instance, we do not abstract from spatial *points* to *topological nodes* such as rooms and corridors, as in [105].

- Among the actions defined for the POMDP model, we have introduced a virtual action, *navigate*, that allows us to put into practice the core principle of the framework and allows to switch from the Cooperation sub-system to the Navigation one. When *navigate* is encountered, the system executes the actions provided by the Navigation path-planner and proceeds towards the destination. The reward function is built to reward when *navigate* is performed in the CS set, that is, when joint intention is ensured between the human and the robot. The system will therefore plan actions to bring the state to a CS state and then perform *navigate* to proceed with the navigation.

- Observations have been defined to estimate the state variables using the sensors available on the robots. We have highlighted how the separation of the Mission into two separate modules precludes the Cooperation POMDP information about which actions, and therefore state transitions, are performed while navigation is active. We addressed this issue by introducing the observation of the action currently executed.

The proposed approach is similar to the one presented by Fiore et al. [35]. The authors present a framework for planning collaborative human-robot tasks. A Situation Assessment module observes the human's activities, position and distance, and estimates his level of engagement to the task. The Collaborative Planner module is an abstract planner which outputs high-level actions and decides whether continue, suspend or abort the task depending on the user's level of commitment. The authors have applied the framework to several applications, including a guide robot.

There are a few differences with our work. The Collaborative Planner in [35] uses hierarchical Mixed Observability Markov Decision Processes (MOMDP), an MDP extension which includes both fully observable and partially observable state variables. The use of POMDPs in our approach allows for a more general method that does not make assumptions on the observability of sensor data, like the human's distance and orientation. The POMDP also accounts for all the required belief management within its

policy generation: beliefs are modeled as probability distributions over states, instead of rule-based predicates.

In the next Chapter, we will describe the experiments carried out to evaluate the implemented framework for the Escort Mission.

Chapter 6

Experiments

In this Chapter we describe the experiments performed to evaluate the proposed approach. Three types of experiments were performed: first on a toy problem, then in a simulated environment, and then with the real robots. We provide and discuss the results obtained for each case.

6.1 Performance Criteria

The evaluation of a collaborative task is not trivial. In this thesis, we have included in the planning process the cooperative aspects of a human-robot shared mission. Most evaluations of guide robots in Literature focus on either the navigation aspect or the user's satisfaction.

Total navigation time may measure how fast a mobile robot reaches its destination; it may be a measure of efficiency for the navigational aspect, but does not evaluate the quality of the cooperation with the human. For instance, a robot would have better navigation times if it did not ensure the human's cooperation and traveled alone.

Surveys that ask users their level of satisfaction about the robot's cooperation, on the other hand, cannot be performed in simulation and cannot be used for comparisons with different methods. They also require a considerable amount of real-world experiments with naive, untrained users that was not possible to perform during the thesis.

In order to evaluate quantitatively the quality of our approach for both the Navigation and Cooperation aspects of the Escort Mission, we introduce the following performance criteria: the *Cooperative Time Rate*, the *Navigation Time Rate*, the *Cooperative Navigation Rate*, and the *Cooperative State Belief*.

The **Cooperative Time Rate** CR measures the ratio of time steps in which the real state belongs to the Cooperative States set, CS, with respect to the total time:

$$CR = \frac{\text{time in } CS}{\text{total time}} = \frac{\sum_t^T \delta(s_t \in CS)}{T}$$

with $\delta(P) = 1$ if P is true, and 0 otherwise. The CR provides a measure of how well the robot ensures the human's cooperation, but not about the efficiency of the navigation task.

The **Navigation Time Rate** NR measures the ratio of steps executing the *navigate* action towards the destination with respect to the total time:

$$NR = \frac{\text{navigate time}}{\text{total time}} = \frac{\sum_t^T \delta(a(t) = navigate)}{T}$$

It is an efficiency measure for the navigation task, but does not give information about the level of engagement of the user.

The **Cooperative Navigation Rate** CNR measures the ratio of *navigate* actions performed in the CS set with respect to the total time. It accounts for both aspects of the cooperative task.

$$CNR = \frac{\text{navigate time in } CS}{\text{total time}} = \frac{\sum_t^T \delta(s_t \in CS, a(t) = navigate)}{T}$$

We additionally measure at each time step the **Cooperative State Belief** ratio BCS, that is the belief of the robot to be in a cooperative state. This measure allows us to estimate the belief error with respect to the real state.

$$BCS = \frac{\sum_t \sum_{s \in CS} b_t(s)}{\text{total time}}$$

We will use these measures as our main performance criteria, along with the total mission time.

6.2 Behavior models

Following Section 5.2.4, we have defined both a movement and an attention model for the human. The attention model is characterized by the probability for changing the Attention Level, as shown in Figure 5.5 and summarized in the following way:

- P_{FD} is the probability of changing from *Focused* to *Distract*
- P_{DF} is the probability of changing from *Distract* to *Focused*
- P_{DL} is the probability of changing from *Distract* to *Lost*
- P_{LD} is the probability of changing from *Lost* to *Distract*
- the probability to stay *Focused*, *Lost* or *Distract* are trivially the complementary of P_{FD}, P_{LD} and $P_{DF} + P_{DL}$ respectively

By changing these values, we can define several behavior models for the human. These models allow us to describe the level of commitment of the human and evaluate the robustness of the policy.

We define the following behavior models:

- AF = *Always focused*: $P_{FD} = 0, P_{LD} = 1, P_{DF} = 1$
- AaF = *Almost always focused*: $P_{FD} = 0.1, P_{LD} = 0.9, P_{DF} = P_{DL} = 0.25$
- MF = *Mostly focused*: $P_{FD} = 0.25, P_{LD} = 0.75, P_{DF} = P_{DL} = 0.25$
- H = *Half-times*: $P_{FD} = 0.5, P_{LD} = 0.5, P_{DF} = P_{DL} = 0.25$
- ML = *Mostly lost*: $P_{FD} = 0.75, P_{LD} = 0.25, P_{DF} = P_{DL} = 0.25$
- AaL = *Almost always lost*: $P_{FD} = 0.9, P_{LD} = 0.1, P_{DF} = P_{DL} = 0.25$
- AL = *Always lost*: $P_{FD} = 1, P_{LD} = 0, P_{DL} = 1$

These models also affect the probability to successfully improve the user's level of attention with the *draw_attention* action. The difficulty increases linearly from *Always focused* to *Always lost*, where the probability to successfully draw the user's attention is half the one for the *Always focused* case.

We did not implement different models for the human motion, described in Section 5.2.4. A set of hand-made probability values were given for the *move to robot*, *stay* and *deviate* human actions, varying according to the Attention level and Proxemic distance. The probability of *move to robot* is higher when the human is *Focused*, except when it is too close to the robot (at *Intimate* the *Focused* person stays on the spot). The probability of *stay* and *deviate* actions, instead, increases with the distance when *Distracted* and *Lost*.

6.3 Grid-World evaluation

The first series of experiments was performed on a simple domain. The aim of these experiments is to compare our approach with different policies generated using state-of-the-art POMDP resolution techniques.

The domain of the experiment is a closed 15 x 15 grid world with no obstacles, shown in Figure 6.1. In the figure, G marks the goal destination, while H and R mark the starting positions of the human and the robot respectively. All these positions are fixed and the same for all experiments.

This series of experiments was performed through a Java program outside of the ROS architecture developed for the COACHES project.

We have run several tests, varying the human's behavior model to evaluate the robustness of the approach. Figure 6.2 shows two example runs of the tests on the Grid-world environment. The blue line describes the human's path and the red line shows the robot's path. Darker lines show when an agent moved along the same path multiple times. Figure 6.2(a) shows a run with an *Always focused* human behavior. The human's actions mostly mirrors the robot's, as he attempts to follow the robot. Figure 6.2(b) shows a run with an *Almost always lost* human behavior. We can see that deviations from the robot's path and back-tracing are more common in the human's path. From

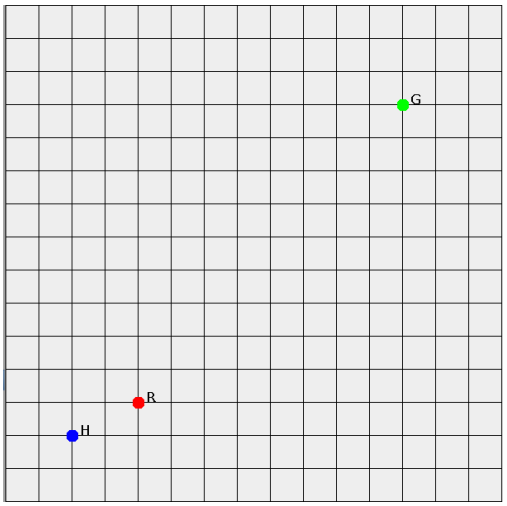
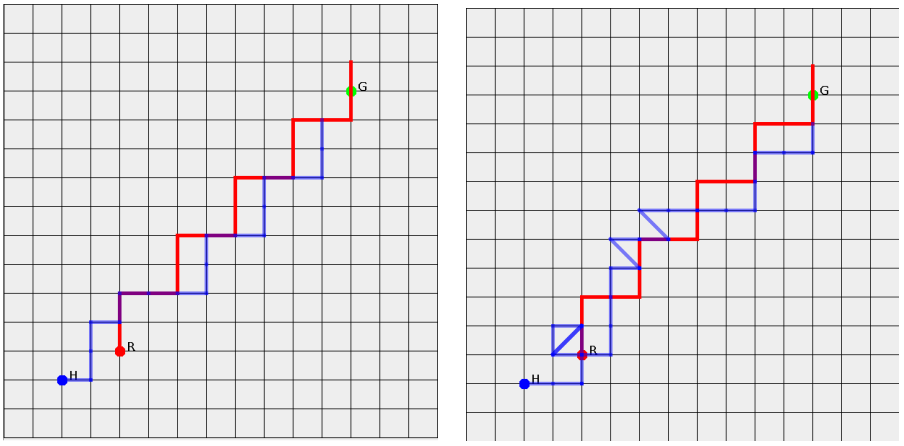


FIGURE 6.1: The grid world environment

the figure it can be inferred that the robot does not change its path, and instead it stays on place and performs *draw attention* actions whenever the human gets away.



(a) Always focused path (b) Almost always lost path

FIGURE 6.2: Trace of the human's and robot's path in the Grid-world environment

In the Grid-world simulations, simple navigation and obstacle avoidance algorithms were implemented. Whenever the human's movements would lead to the position of the robot or beyond the map's boundaries, the human's action (*move to robot*, *stay* or *deviate*) would be drawn randomly until it leads to a valid state. The experiments terminated whenever the human was close to the goal.

Because the human's movements and attention are stochastic, for each behavior model we have run 30 tests and considered the average results.

6.3.1 Performance results

We used the Grid-world experiments to compare our results with state-of-the art POMDP resolution techniques. In particular, we compared our policy generation approach, trough the resolution of a Discrete-BMDP, with an approximate POMDP solver. We have used the `pomdp-solve` software [1] to solve the Cooperation POMDP. The *finite grid* algorithm [91] was chosen, using 5053 belief points. This method was chosen since it allowed to specify the number of belief points to use and set it as close as possible to the 4800 points generated by the DBMDP (Section 5.2.6). The finite grid policy was computed with a rolling horizon 5, meaning that the same policy for the next 5 steps is applied at every time step. The uniform distribution on S was used as starting belief.

The finite grid algorithm, however, could only solve an incomplete version of the Cooperation POMDP model. Full observation functions depending on both the input and output state, $O(s, a, s')$, are not supported by the `pomdp-solve` software. A partial, incomplete observation function $\tilde{O}(a, s')$ for the Cooperation POMDP, which does not account for the observation of actions performed during navigation, was used instead. Hence, through the following experiments we will compare the results obtained from:

- the policy generated by a DBMDP model with a *full* observation function
- the policy generated by a DBMDP model with a *partial* observation function
- the policy generated using the *finite grid* algorithm on a POMDP with a partial observation function

6.3.1.1 Full model results

Figure 6.3 shows the average Cooperation Rate, the Navigation Rate and Cooperative Navigation Rate of the full model by varying the human's attention model.

With the *Always Focused* behavior, the human always stays focused and close to the robot, therefore the Cooperation rate is 1. As the behavior model is less committed, the CR and thus the CNR decreases as expected. The navigation rate decreases as well, since the robot spends more actions to re-establish cooperation instead of navigating. In

[1] `http://www.pomdp.org/code/index.html`

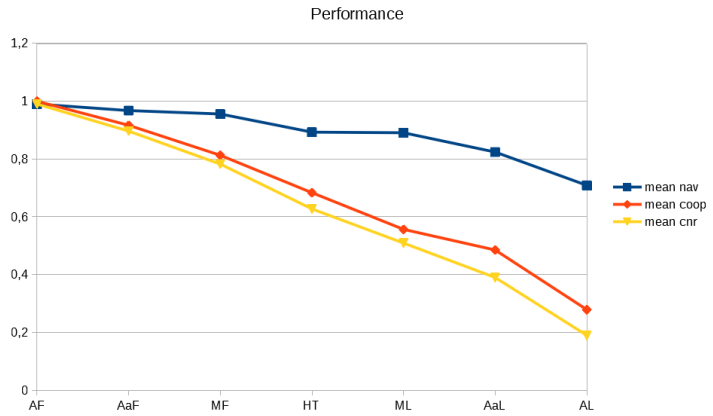


FIGURE 6.3: Performance in grid-world: full model

the *Always Focused* case, which is the ideal situation, no action is required for ensuring the joint intention and thus the NR is close to 1. Remark that even in the *Always Lost* case, most actions performed by the robot are navigation actions. This is due to the belief error of the robot with respect to the real state. Figure 6.4 shows the average belief to be in a cooperative state, *BCS*, compared with the real cooperation rate *CR*. The robot believes to be in a cooperative state more often than the actual frequency of a real cooperative state, therefore it performs *navigate* actions even when cooperation is missing. The lowest belief error is found with the *Mostly Focused* case, which is actually the planning model: the behavior model, and the corresponding set of probabilities, used during the planning phase. The belief error is hence at its minimum when the execution model coincides with the planning model, as we would expect.

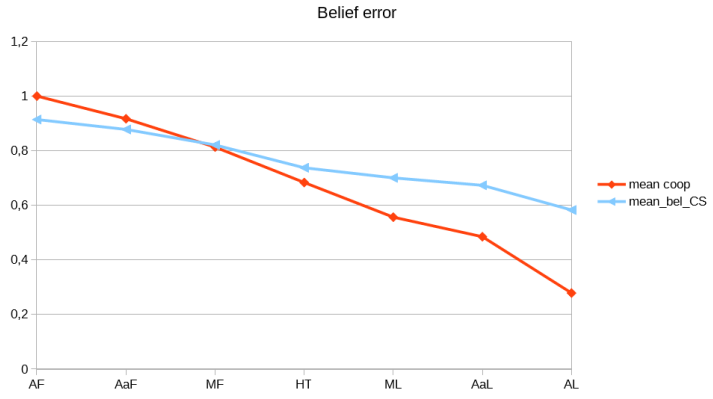


FIGURE 6.4: Cooperation Belief in grid-world: full model

Lastly, Figure 6.5 shows the average total execution time over 30 simulated tests, using the full model policy. As expected, execution time increases with less committed behavior models. The more the human is in a Distracted or Lost attention level, the more he

stays on the spot or deviates in a random direction instead, thus taking more steps to reach the destination.

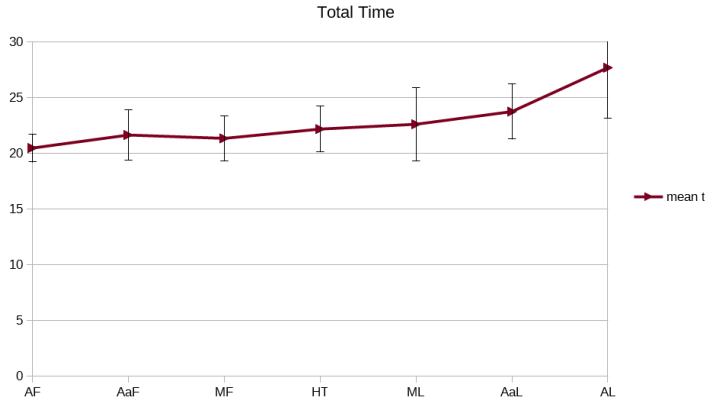


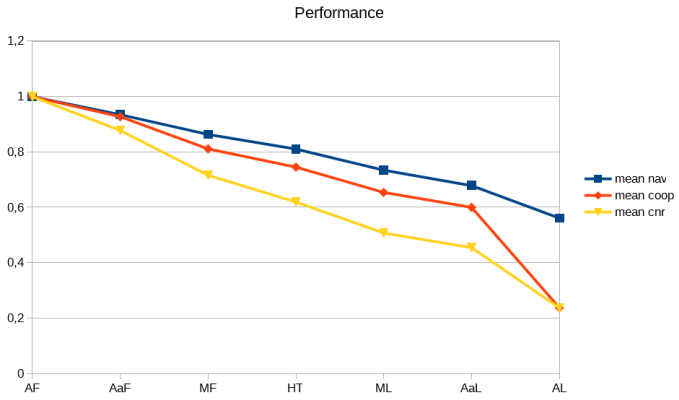
FIGURE 6.5: Execution time in grid-world: full model

6.3.2 Partial model and Finite Grid results

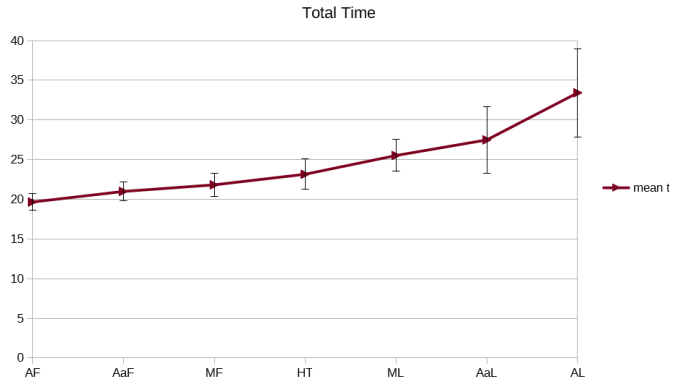
We present now the results obtained using the model with the partial observation function. Figure 6.6 shows the average performance and execution time depending on the human behavior model. Similarly to the full model, performance values (Figure 6.6(a)) are close to 1 for the ideal *Always Focused* case and decrease accordingly with the lack of human commitment. Execution time (Figure 6.6(b)), instead, increases as exepected. While with more focused behaviors the human follows more often the robot and is more predictable, with more distracted ones the human has an higher probability of deviating randomly, thus increasing the standard deviation for the average execution time.

The analysis of the cooperative state belief, shown in Figure 6.7, proved interesting. As we can see, the average belief to belong in a CS state is lower than the real cooperation rate, and is minimal in the *Almost always Lost* case, despite the fact that the *Mostly Focused* set of probabilities was used for the planning process of the partial model. The incomplete observation function induces a loss of information during navigation. On the other hand, when the human has a low commitment to the task, there is a high chance that he may wander outside of the robot's field of view, resulting in a loss of information for the robot.

As previously mentioned, the finite grid policy was computed using the incomplete observation function. However, the results from executing it are quite different from the



(a) Performance



(b) Execution time

FIGURE 6.6: Performance and execution time in the grid-world simulation using the partial model

partial DBMDP model results. Figure 6.8 shows the performance and execution time results, while Figure 6.9 shows the cooperation rate and cooperation belief. As we can see, the minimal belief error is within the *Mostly Focused* case, which is more in line with the full DBMDP model. The incomplete observation function has less impact on the belief error because the finite grid policy does not execute pre-computed discretized belief transitions, and instead computes the belief update during execution.

6.3.3 Comparison

The very first term of comparison we describe the planning time. Planning using the DBMDP approach required about 45 minutes to compute the policy, and another hour to convert the DBMDP policy into a POMDP policy [2]. The finite grid policy required

[2] on a 64 bits Intel i5-3570 CPU machine with 3.40GHz and 8.2GB RAM

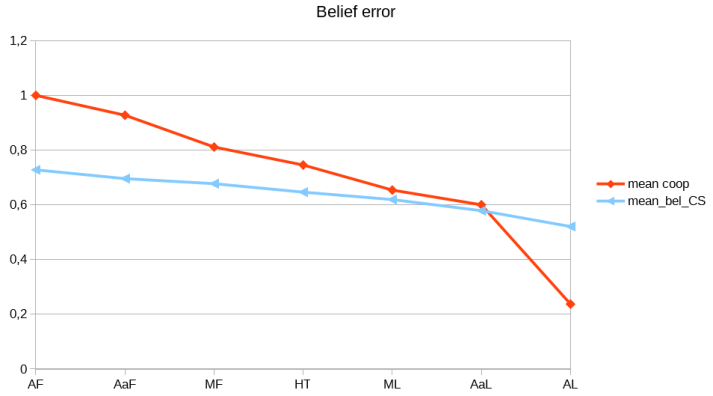


FIGURE 6.7: Cooperation Belief in grid-world: partial model

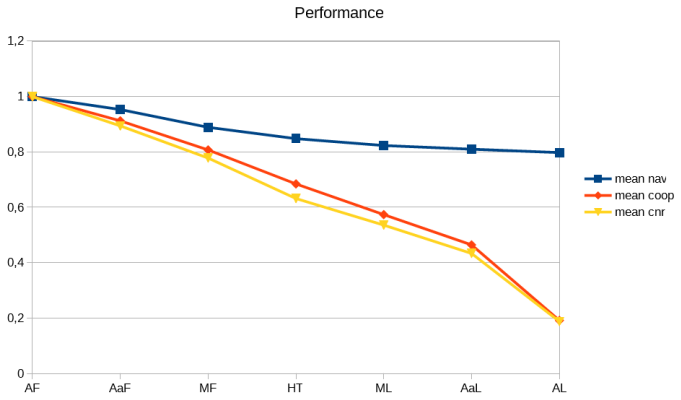
35 minutes to compute [3]. While the finite grid policy was computed with a fixed horizon 5, the DBMDP policies were computed on an infinite horizon and computation stopped after 53 iterations.

The comparison of CNR performance between the three models is shown in Figure 6.10. It can be seen that there is no significant difference between the models used, which may seem surprising considering the loss of information related to the incomplete observation function. We can notice that the partial model does have a lower performance on the *Mostly focused* case and an higher value at the *Almost always lost* case, which is coherent with the belief error results previously shown. More insight, however, may be gained by looking at the Navigation Rate and Cooperation Rate (Figures 6.11 and the 6.12).

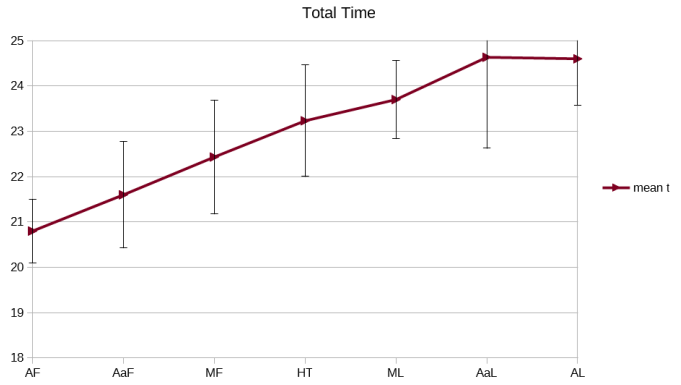
Figure 6.12 shows that the partial policy has higher cooperation rates with respect to the full model and finite policies, which have close values. Regarding the Navigation rate, however, the partial policy executes consistently less *navigate* actions, the full policy being the one that performs more navigation in most behavioral cases. Because the CNR metric takes both aspects into account, the net result for the partial model is close to the full and finite grid policies.

The partial policy, however, has arguably the worst performance results. Even if it ensures a better cooperation for the human, it does so at the expense of performing less navigation actions. This phenomenon is confirmed by the execution time comparison (Figure 6.13), showing that the partial model policy takes more time to reach the destination.

[3] on a 4 AMD Opteron 6174 CPU machine with 2.20GHz and 256GB RAM



(a) Performance



(b) Execution time

FIGURE 6.8: Performance and execution time in the grid-world simulation using the finite grid policy

Figure 6.14 compares the belief error among the models, computed as $|CR - BCS|$. While the incomplete observation function significantly affects the belief error for the partial model, it has less impact on the finite grid policy.

The objective of the grid-world simulations was to compare our policy generation approach with a well-known approximate POMDP resolution technique. The experiments we have performed show that the two methods provide close results on most metrics. In the following experiments, performed in a more complex simulation environment, we couldn't execute the rolling horizon *finite grid* policy, since it was incompatible with the Petri Net Plan architecture. Because the evaluation of the finite grid policy proved similar to the full model policy in the grid-world experiments, we could proceed with further tests using our approach, and expecting the finite grid policy to provide results similar to those shown in the following sections.

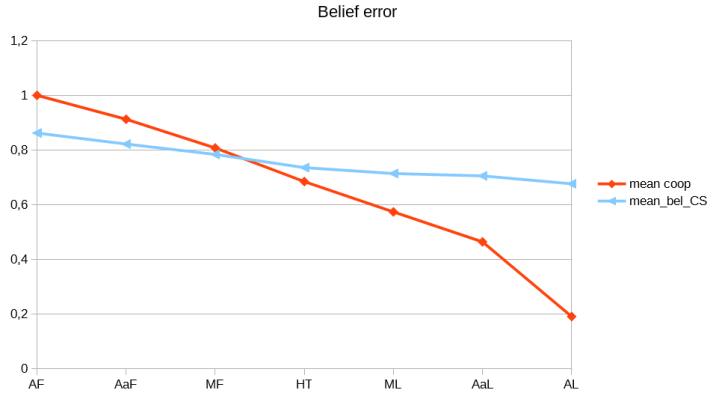


FIGURE 6.9: Cooperation Belief in grid-world: finite grid policy

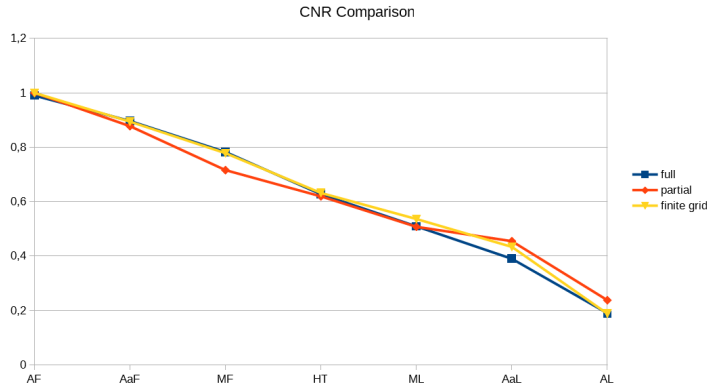


FIGURE 6.10: CNR performance results comparison

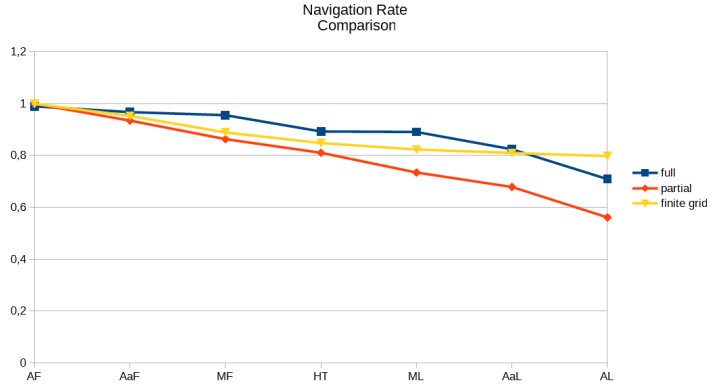


FIGURE 6.11: Navigation rate comparison

6.4 Simulated Environment

The second set of experiments was performed in a more complex simulated environment. We used the Stage [4] software to simulate a section of our lab. Stage can be interfaced with ROS, which is the middleware adopted in the real robots. As such, most of the

[4] `http://wiki.ros.org/stage`

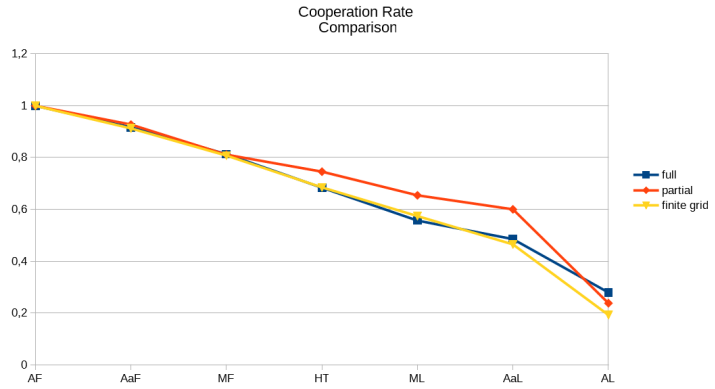


FIGURE 6.12: Cooperative rate comparison

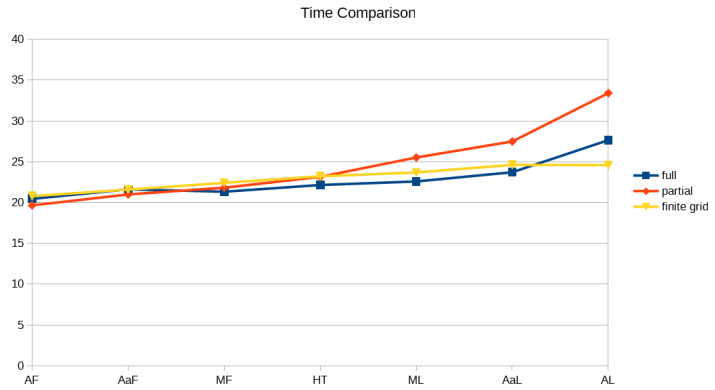


FIGURE 6.13: Execution time results comparison

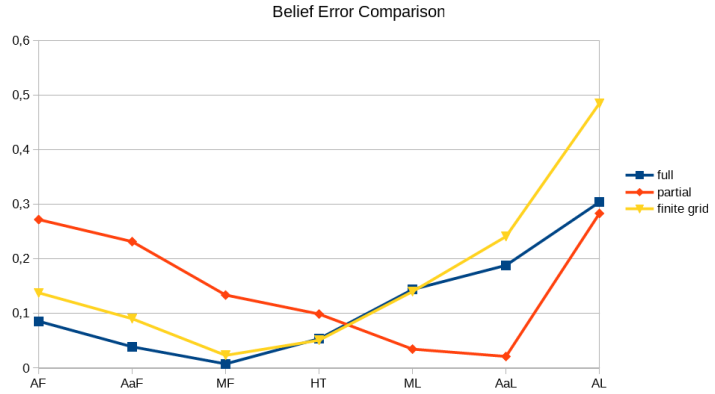
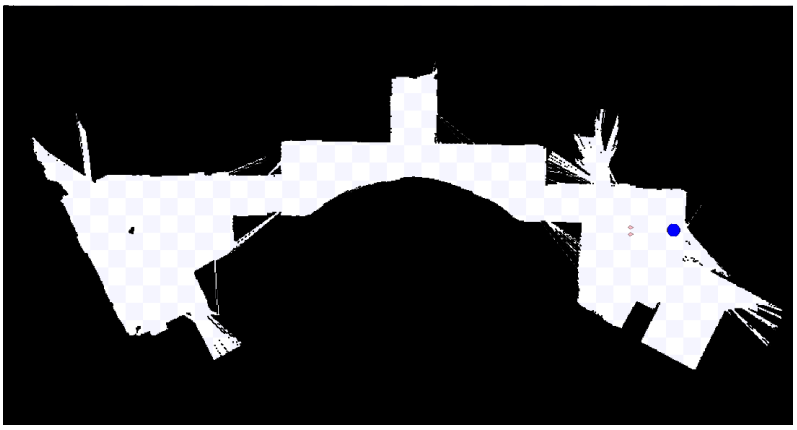


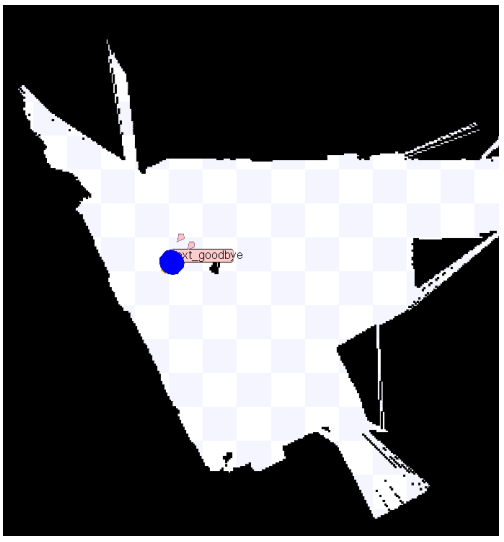
FIGURE 6.14: Belief error comparison

modules in the COACHES software architecture are executed in the Stage simulation. Because there is no camera video processing in the simulated environment, the Situation Assessment module was slightly modified to provide the required observations using the Stage's environment data (see Figure 5.6). Specifically, the Attention level of the person is established by checking his orientation with respect to the robot, instead of using the camera's face detection technique.

In the simulation environment, shown in Figure 6.15(a), the blue circle represents the robot, and the human is represented by his feet. This allows the simulation of person detection using the laser scans.



(a) Initial position



(b) Goal position

FIGURE 6.15: A section of our lab simulated in Stage. The blue circle represents the robot, and the pair of feet represents the human.

For each behavior model, 10 simulations have been performed, using the same goal and starting positions for the human and the robot. Each simulation terminated whenever both human and robot reached the destination within a cooperative state, or after 5 minutes of system clock time.

Figure 6.16 shows the Navigation Rate, Cooperation Rate and Cooperative Navigation Rate of the full model policy varying the human behavior model. The implementation on Stage resulted in lower performances with respect to the grid-world simulation. Even with a *Always Focused* behavior, the system is in a cooperative state only about 70% of

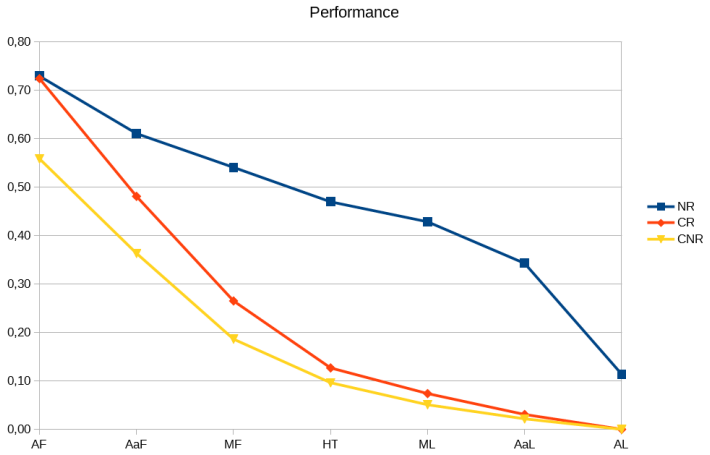


FIGURE 6.16: Performance results in the Stage simulation

execution time. On the other hand, only 70% of the robot's actions consist of *navigate* actions, meaning that the remaining actions are performed to re-establish the missing cooperation. This is due to the difference between the robot's and human's speed. In the current implementation, the robot moves faster than the human when navigating. Speed may vary according to the presence of obstacles, but does not adapt to the user's pace. The robot also performs several turns which may put the human within the sensors' blind spots. Therefore, the robot regularly stops and turns around to search for the guided person when beyond the range or field of view of its sensors. This phenomenon did not show in the grid-world experiments because of the small world size and discrete nature of the environment, time and actions.

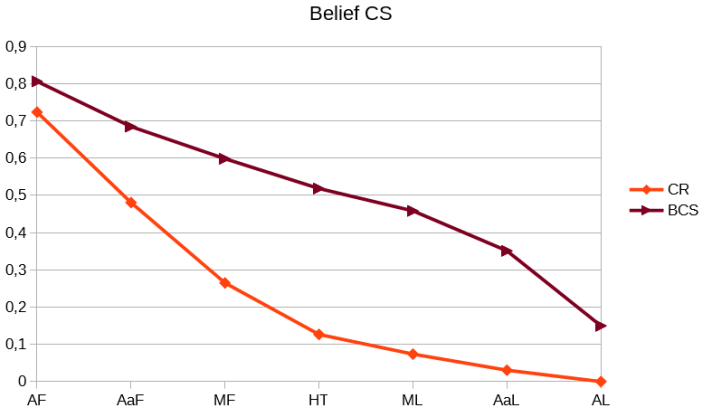
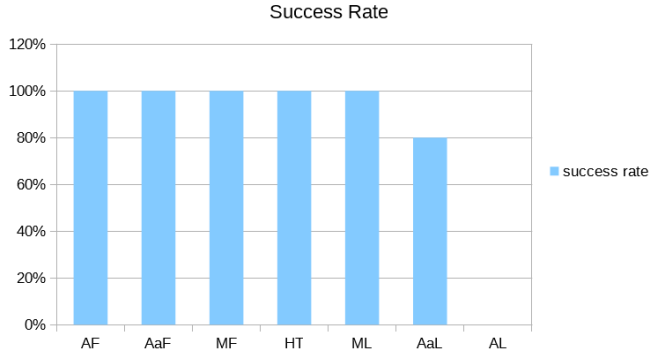


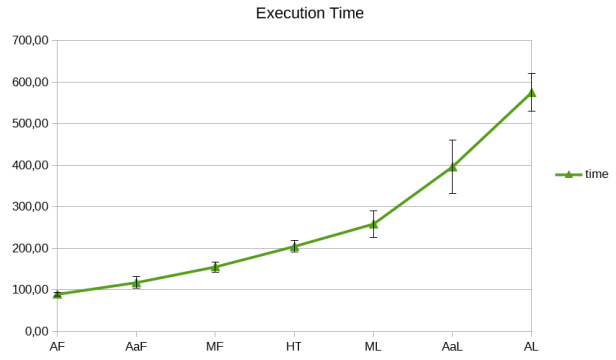
FIGURE 6.17: Belief error in the Stage simulation

As the behavior varies, the person becomes more distracted and stays less frequently behind the robot. The performance results are significantly affected by the number of stopping and turning actions required to re-establish cooperation: in the *Always Lost*

case, the system is actually never in a cooperation state. Figure 6.18(a) shows the success rate of the experiments of each behavior model: the percentage, of tests that terminated with the arrival of both human and robot at destination. Otherwise, the test terminated after 5 minutes of clock time. Despite the low CNR performance values, the robot always managed to guide the human to the destination for most behavioral models. Two failures occurred with the *Almost Always Lost* case; in the *Always Lost* case the robot always failed to guide the human. While the success rate proves the robustness of the approach, the number of time steps required to achieve the objective does however increase with lower levels of human commitment, as shown in Figure 6.18(b).



(a) Success rate



(b) Execution Time

FIGURE 6.18: The average execution time and success rate of experiments in the Stage simulation.

6.5 Test on Real robots

This Section describes the preliminary experiments carried on the Cadomus robot in our lab. Development on the real robots is an ongoing process and more recent tests are planned to be performed at the moment of the thesis writing.

Currently, the cameras are able to detect the position, distance and orientation of a person with the aid of test patterns. We plan on soon performing further tests using online face tracking techniques to detect such features without the need of test patterns. Through the GUI, the robot first offers to provide assistance and suggests a list of possible destinations. Once the user selects the destination, the Escort Mission starts. Whenever the robot finds itself in a cooperative state, it starts the navigation module and performs path-planning towards the goal. As long as the human is detected by the rear or front camera and follows within a Social distance, the robot proceeds with the navigation until the goal is reached (Figure 6.19(a)). We have tested the robot's reaction to two main situations:

- Whenever the human stops looking at the robot, but is still detected by a camera, the robot will use the speech system to draw its attention by inviting him to keep following. (Figure 6.19(b)).
- When the cameras stop detecting the person, the robot stops its navigation. It then starts turning around hoping to detect the human with its front or rear cameras. Once the user is detected again, the robot restarts the navigation module. (Figure 6.19(c) and 6.19(d))

Once the destination is reached, the mission ends and the robot waits new users next to the entrance.

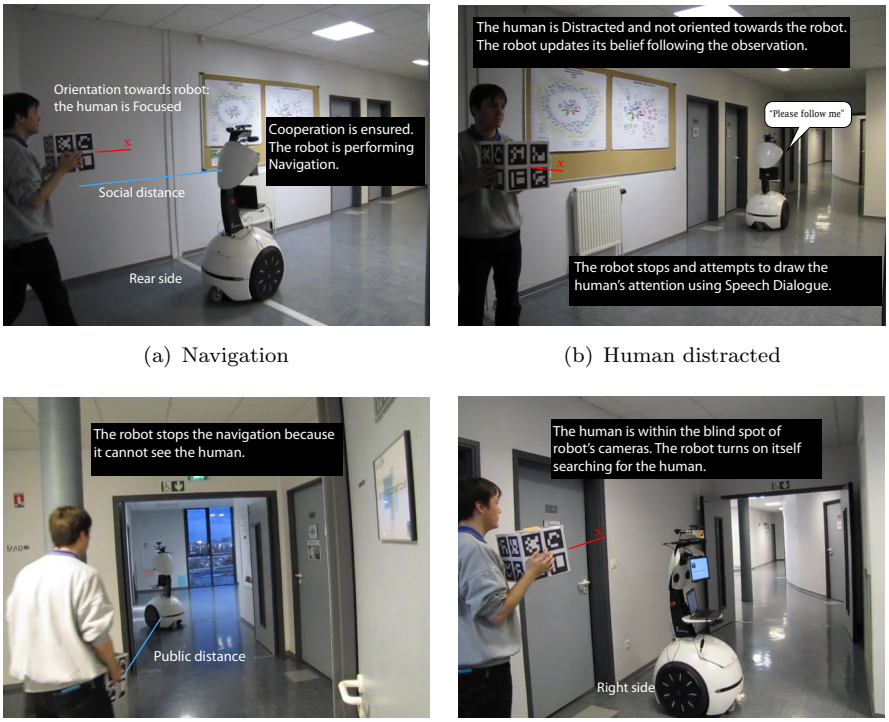
The full video of the demonstration, showing the described behaviors, is available online[5]

6.6 Chapter conclusions

In this Chapter we have defined performance criteria to evaluate both the Cooperation and Navigation aspect of an Escort Mission. To our knowledge, most guide robots in Literature are mainly evaluated trough user feedback and lack of objective metrics to evaluate how the robot ensures cooperation in an human-robot joint activity.

We have defined the Navigation Rate, Cooperation Rate and Cooperative Navigation Rate, as well as the Cooperative State Belief metric. While the CNR metric attempts

[5] `https://youtu.be/r2ZizBcczGY`



(a) Navigation

(b) Human distracted

(c) No detection

(d) Searching for person

FIGURE 6.19: Examples of situations arisen during the experiment with the Cadomus robot.

at taking both aspects of the Mission into account, all measures should be examined for an accurate and throughout analysis of the system's performance.

We have carried several sets of experiments with different objectives. In the first set of experiments, we have compared our approach with a state-of-the art planning algorithm. Specifically, we have compared the policy generated by a Discretized Belief-MDP and translated into a POMDP policy trough the belief-shift function, with an approximate policy directly generated from the POMDP model. The results showed that our approach does not differ significantly from the finite-grid policy.

The grid-world experiments also showed the impact of information loss when the robot is performing navigation, and motivates our solution to use a full observation function on both input and output states, $O(s, a, s')$, and to use observations on the executed action to help estimate the current state of the system.

All of our experiments were performed using different behavior models for the human. Therefore, we could evaluate the robustness of the policy with respect to varying degrees of commitment to the shared task.

The set of experiments in the Stage environment made use of the COACHES architecture properly and was meant to be a more accurate simulation of a real-life execution. As such, we could compare the performance results of execution in a larger, more complex and continuous map with the smaller, discrete and toy-like domain of the grid-world simulations. The results highlighted the importance for a guide robot of adapting the speed to the human's pace. Despite the worsening of performance and longer execution times, the success rate proved the robustness of our approach in most behavioral models.

Part IV

Conclusions

Chapter 7

Conclusion and Perspectives

7.1 Synthesis of contributions

In this thesis, we investigated the challenges of decision-making for service robots in public spaces. Specifically, we have addressed the possible lack of engagement that users may have when performing joint activities with the robot. To this end, we have proposed a decision-making framework capable of providing plans for an human-robot joint activity that may ensure the human's cooperation.

We have presented the challenges that may arise when cooperating with users in an unrestricted and dynamic environment, and decided to base our approach on POMDPs to handle these challenges.

We have developed a novel framework for planning human-robot cooperation, based on two principles: a hierarchical structure built on state abstraction, and task decomposition. While these principles have already been used in Literature for achieving efficiently the Task aspect of a shared Mission, their use in the thesis is more oriented towards the Cooperation aspect. The main idea of the proposed framework is the separation of the overall Mission into two separate planning modules that can be solved independently.

We also have presented our own approach for solving the Cooperation POMDP model. This approach consists in formalizing the POMDP as a Discrete Belief MDP, and solve it trough classic MDP resolution techniques. In order to execute the resulting policy as

a POMDP, we have introduced the *belief-shift* function, which re-introduces the dependency on observations in the transition function of the DBMDP. This approach allows us to reduce the planning complexity and generate offline, infinite horizon policies easily implemented as Finite State Machines for the Cooperation aspect.

Our framework was evaluated trough the Escort Task application scenario. We have defined performance criteria to measure the quality of both aspects of the Escort Mission, navigating towards a destination and ensuring that the guided user follows the robot throughout the task execution. We used these criteria to compare our policy with an off-the-shelf POMDP resolution technique, proving the validity of our approach. The robustness of our framework was evaluated by varying the level of engagement of the simulated human to the shared task. Finally, the framework was successfully implemented on real robots.

7.2 Perspectives

There are several directions along which this thesis work can be further developed and that could be investigated.

- The primary direction of development is the extension to Group Escorts. Section 5.4 briefly presented how the framework could be adapted to ensure the cooperation of multiple users. This extension, however, still require further investigation and an actual implementation.
- The experimental results highlighted the need for a speed adaption mechanism. Additional actions, such as *slow down* and *accelerate*, can be added to the POMDP model, so that the planning process decides how to manage the robot's speed depending on the user's pace. Adding these actions, however, may not be trivial since they would affect the navigation part of the framework and the state-space of the POMDP.
- More experiments and comparisons could be performed. Our DBMDP-based policy can be compared to other state-of-the-art algorithms, such as PBVI or HSVI. In order to validate the proposed approach, which separates Task and Cooperation aspects of a joint activity, we would need to compare the results with an unified

model. While the difference in state-space size and planning complexity is evident, the difference in performance remains to be evaluated.

- Currently, the robot attempts at re-establishing cooperation with the user when missing. A desired feature would be the capability to understand when the user wishes to abort the task. The Mission Status layer of our framework should differentiate between a temporary missing joint intention and a lack of commitment, and thus decide whether or not abort the mission.

- The probability values used for the transition function of the Cooperation POMDP, in particular for the human attention and movement models, are currently handmade. More accurate values could be obtained by integrating the model with a learning process and collecting data with real, untrained users.

- We have introduced a factored discretization method to reduce and customize the generation of belief points. We believe that the factored representation of belief states, however, could be further exploited by POMDP resolution techniques

- Finally, the proposed framework is designed to be flexible, and is not limited to the Escort Mission but could be applied to different HRI tasks.

Appendix A

Modèles Décisionnels pour la Coopération Homme-Robot dans les Activités Jointes

A.1 Introduction

Le déploiement de robots de service conçus pour fournir assistance aux êtres humains dans des espaces publiques tels que des centres commerciaux, des musées, ou des aéroports, est une tendance en forte croissance. Ce type d'applications présente cependant plusieurs difficultés pour le robot. Dans le cadre de cette thèse, nous allons nous concentrer sur les difficultés liées à la coopération homme-robot dans une activité jointe du point de vue de la planification et de la prise de décisions. Plus précisément, nous allons développer des modèles décisionnels qui puissent prendre en compte l'imprévisibilité du comportement humain et un éventuel manque d'engagement de la part de l'humain à coopérer avec le robot. Tandis que plusieurs applications assument un niveau persistant d'engagement de l'humain à accomplir une tâche jointe avec le robot, cette assomption n'est pas toujours vérifiée dans les espaces publiques. Dans un lieu publique, les utilisateurs des robots de service sont pour la plupart des passants qui peuvent facilement être distraits par l'environnement dynamique et qui peuvent facilement abandonner le robot. La planification de la tâche jointe doit donc tenir compte de l'évolution du niveau d'engagement de l'homme envers la tâche à accomplir. L'objectif de la thèse est donc le développement

d'un modèle décisionnel qui essaye de maintenir la coopération de l'humain tout au long de la tâche jointe.

La thèse a été menée dans le cadre du projet européen COACHES. L'objectif du projet est le développement de techniques permettant de concevoir un robot de service pour accueillir et assister les clients d'un centre commercial. Ce scénario constitue le principal exemple d'application pour la thèse. Plus précisément, nous nous sommes intéressés à la tâche d'Escorte, où le robot offre de guider un client du centre commercial à une destination désirée. Nous considérons l'escorte comme une tâche jointe où la personne et le robot doivent coopérer pour atteindre ensemble le but. La personne peut être distraite par les magasins, ou par des événements imprévus, et le robot doit s'assurer que la personne soit bien en train de le suivre.

A.2 État de l'art

A.2.1 Robotique de service dans les espaces publiques

L'Interaction Homme-Robot (HRI) est un domaine robotique en forte croissance depuis la dernière décennie. Plusieurs applications et études ont été menées à ce sujet dans plusieurs domaines : intelligence artificielle, fusion de données, apprentissage, langage naturel, ainsi que psychologie et sciences sociales. Des études des caractéristiques principales de l'HRI, les domaines afectées et des exemples d'applications se trouvent dans [3], [4] et [5].

En particulier, les robots de service sont de plus en plus déployés dans des espaces publiques, tels que des musées [7][9][11][12][13], des centres commerciaux [14][16][17], des aeroports [19] et des maisons de retraite [18].

A.2.1.1 Défis des espaces publiques

Les espaces publiques présentent plusieurs difficultés pour les robots:

- **Environnement dynamique**: l'environnement est imprévisible et peut changer à tout moment. La présence d'une foule peut compliquer l'activité du robot,

tandis que ses capteurs doivent faire face à des occlusions dynamiques et à un haut niveau de bruit. Le modèle décisionnel du robot doit donc être suffisamment robuste pour réagir rapidement et efficacement à des changements imprévus dans l'environnement.

- **Navigation sociale**: pour qu'il puisse opérer de façon naturelle dans un espace publique et être socialement acceptés par les utilisateurs, le robot doit respecter les conventions sociales qui régissent les interactions entre humains. En particulier, le robot doit respecter les distances sociales associées au confort et à l'acceptation sociale de chaque personne. L'étude de ces distances a été introduite par Hall [20] et elle est connue sous le nom de Proxémie. Hall définie quatre espaces sociaux centrés sur une personne (Figure A.1): *Intime* (entre 0 et 46 cm.), *Personel* (entre 46 et 122 cm.), *Social* (entre 1.2 et 3.7 m.), et *Publique* (entre 3.7 et 7.6 m., et plus).

- **Détection et tracking**: pour pouvoir interagir avec une personne, le robot doit d'abord la détecter. Plusieurs applications implémentent en outre le suivi de la personne, de sa position, de sa distance et de son orientation. Ces données peuvent fournir des informations sur l'activité de la personne et sur son état mental, ainsi qu'améliorer la sociabilité du robot. Plusieurs méthodes ont été développées pour détecter et suivre un utilisateur: en fournissant aux utilisateurs des marqueurs RFID (Radio-Frequency IDentification) [14], en détectant les pieds des personnes à partir des capteurs laser [25][26], en identifiant des caracteristiques visuelles à partir des caméras vidéo [27] ou en utilisant des caméras de profondeur [28].

- **Dialogue**: la communication est une partie essentielle des interactions homme-robot, spécialement pour les activités jointes. La communication verbale, les gestes, les écrans et les expressions faciales peuvent être utilisés pour communiquer avec l'humain; inversement, de la même façon, le robot peut utiliser ces moyens pour estimer l'activité de l'humain, son niveau attention et ses intentions.

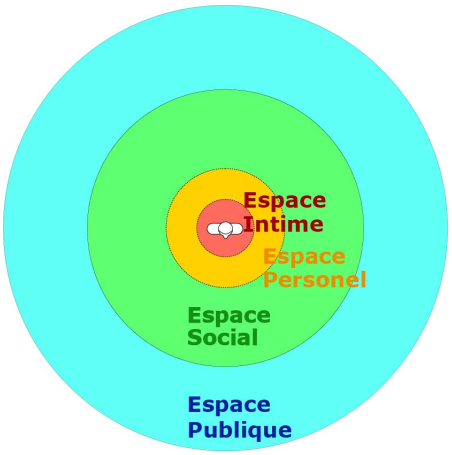


FIGURE A.1: Distances sociales en Proxémie

A.2.1.2 Robots guides

Dans la robotique de service en lieu publique, l'une des applications les plus populaires est celle du robot guide. La tâche d'Escorte du projet COACHES, utilisée comme application des travaux de cette thèse, n'est que la dernière d'une longue série de robots guides commencée par Rhino [7] et Minerva [1], et dont la plupart a operé dans des musées. Très rapidement le développement des robots guides s'est concentré sur l'aspect interactif avec les utilisateurs. Le robot Chips [13] attire l'attention des visiteurs en performant des mouvements prosodiques pendant les longues présentations statiques dans le musée. RoboX [32] est doté de reconaissance vocale, de détection et de suivi de personnes, ainsi que d'une matrice LED utilisée pour exprimer ses emotions. Urbano [8] utilise aussi des gestes des mains et un visage robotique pour s'exprimer.

Plus récemment, des travaux ont essayé de tenir compte de l'imprévisibilité du comportement humain. Le robot Jido [12] essaye de comprendre le niveau d'engagement de l'utilisateur humain. Il calcule une prédiction de sa trajectoire pour estimer ses intentions et pour obtenir une navigation fluide et socialement acceptable. Zhang [34] propose d'utiliser des champs potentiels artificiels pour que le robot s'adapte aux comportements non-cooperatifs de l'humain. Fiore et al. [35] décrivent une approche qui permet d'estimer l'engagement de l'utilisateur et de décider si d'adapter la vitesse du robot, de suspendre temporairement la navigation ou de l'abandonner. Ces travaux témoignent l'intérêt de la recherche à modéliser la coopération humaine et son niveau d'engagement dans la planification pour une tâche jointe.

A.2.1.3 Coopération, Intentions et Attention

Plusieurs travaux ont proposé des modélisations de la coopération entre agents. Une revue des principes et des caracteristiques les plus importantes d'une coopération homme-robot est faite par Bütepage et Kragic dans [39].

La modélisation d'une coopération entre agents consiste avant tout en une formalisation de l'état mental de chaque agent. L'architecture BDI (Belief, Desire, Intention) [42] est une approche populaire qui modélise un agent rationel avec un ensemble de croyances que l'agent a sur l'état du monde, de désirs que l'agent veut voir satisfaits, et d'intentions, qui sont des désirs que l'agent s'est engagé à accomplir. Une vrai coopération, cependant, est plus que la somme des actions des agents [38]. Cohen et Levesque ont donc introduit une extension au modele BDI, appelé Joint Intention Theory [40][41]. Ce formalisme décrit l'état mental des agents lors d'un travail d'équipe en utilisant un langage basé sur la logique propositionnelle et sur des opérateurs temporels. Pour qu'une équipe puisse coopérer de façon efficace, ses membres doivent partager un état mental appelé Intention Jointe, défini comme *un engagement joint à accomplir une action tout en croyant mutuellement accomplir l'action en tant que équipe au cours de l'exécution.*

En conséquence, chaque agent doit être capable d'estimer l'état mental des co-equipiers, ce qui n'est pas évident avec les êtres humains. Plusieurs travaux ont développé des approches pour estimer les intentions de l'humain. Certains essayent de prédir la trajectoire d'une personne en observant sa vitesse, ses mouvements et d'autres caractéristiques [48][34] ou en modélisant des champs potentiels [49]. Le fait que une personne se rapproche d'un objet ou d'une zone spécifique peut révéler ses intentions [50]. D'autres travaux utilisent des Machines à Etats Finis (FSM) [51] ou des Processus Décisionnels de Markov (MDP) [52][36] pour modéliser et apprendre le comportement humain.

Cependant, dans le cadre de cette thèse, nous nous focalisons sur l'estimation de l' *attention* de l'humain. L'attention d'une personne concerne l'entité qui est actuellement l'objet de son intérêt, ou l'absence d'une telle entité. Les principaux indices pour comprendre quel est l'objet de l'attention d'une personne sont l'orientation de sa tête et de son regard. Ces indices ont été utilisé pour permettre à un robot de s'addresser à une personne [54] et pour comprendre quelle personne est en train de s'adresser au robot [53]. Le champ d'attention d'une personne est un sous-ensemble de son champ de vision [55]: ce n'est pas ce que la personne peut voir, mais où elle est en train de poser son

regard. Bien que l'attention soit strictement liée au concept de intention [57] et qu'elle puisse être utilisée pour inférer l'intention d'un humain ou d'un robot [29], il s'agit de deux concepts différents. L'attention n'est pas nécessairement liée à une tâche, et elle peut être focalisée nulle part. Les utilisateurs peuvent être distraits par l'environnement et abandonner l'interaction avec le robot. Ceci est particulièrement vrai dans les espaces publiques ou le robot doit mettre en place des mécanismes spécifiques pour maintenir l'attention de l'utilisateur, comme montré dans [2].

A.2.2 Modèles de planification

Dans cette section, nous présentons les principaux modèles de planification. La planification est l'acte de générer un plan, c'est à dire une séquence d'actions effectué par un agent dans le but d'amener le système à un état désiré. Le système consiste en un ensemble d'agents et l'environnement dans lequel ils se situent. Un état est une description d'une configuration possible du système. L'agent change l'état du système en effectuant des actions. Une transition est un passage d'un état à l'autre à travers une action. Une transition est dite déterministe si elle amène toujours au même état d'arrivée en effectuant la même action depuis le même état de départ. Dans le cas contraire, la transition est dite stochastique et peut amener à un ensemble d'états, auquel elle associe une distribution de probabilités. Un système est dit partiellement observable si l'état courant du système n'est pas toujours connu, ce qui peut être dû à des imprécisions des capteurs, des occlusions, etc.

Table A.2.2 résume les différents types de systèmes et les principaux modèles décisionnels de la Littérature.

	Complètement Observable	Partiellement Observable
Déterministe	Planification Classique	PKS
Stochastique	MDP	POMDP

TABLE A.1: Modèles de planification

Dans le cadre de la planification classique, STRIPS [59] est l'un des formalismes les plus populaires. Il s'agit d'un langage qui décrit les états sous forme de prédicats, et qui

génère les plans grâce à un démonstrateur de théorèmes. Dans le cadre partiellement observable, PKS [62] est une généralisation de STRIPS qui décrit les croyances des agents sur l'état du monde plutôt que l'état en soi.

Le domaine d'application de cette thèse étant très dynamique et imprévisible, surtout en considérant le comportement humain, nous nous focalisons sur les modèles de planification probabiliste.

A.2.2.1 Processus Décisionnels de Markov

Les Processus Décisionnels de Markov (MDP) [64] sont un modèle efficace de planification stochastique.

Définition *(MDP)* A.1. Un MDP est processus stochastique contrôlé, défini comme une tuple $< S, A, T, R, H >$, où:

- S est un ensemble discret et fini d'états s;
- A est un ensemble discret et fini d'actions a;
- $T : S \times A \mapsto \Pi(S)$ est une fonction de transition probabiliste, telle que $T(s, a, s') = Pr(s'|s, a)$;
- $R : S \times S \times A \mapsto \Re$ est une fonction de récompense;
- H est l'horizon de planification;

Les MDP possèdent la propriété de Markov [65], c'est à dire, à chaque instant t

$$Pr(s_{t+1}|s_0, a_0, s_1, a_1, ..., s_t, a_t) = Pr(s_{t+1}|s_t, a_t)$$

Résoudre un MDP signifie trouver une politique π, c'est à dire une fonction qui associe à chaque état s du MDP une action a à exécuter. Les politiques sont évaluées selon une fonction de valeur V. Pour un modèle à horizon fini, la fonction de valeur d'une politique π pour un état s est communément définie comme la somme des récompenses espérées en suivant la politique π pour les prochaines H étapes :

$$V_H^{\pi}(s) = E\Big[\sum_{t=0}^{H-1} r_t\Big] \quad \forall s \in S$$

Pour un horizon infini, la fonction de valeur est atténuée par un facteur γ, qui privilégie les récompenses obtenues à court terme dans l'horizon.

$$V^{\pi}(s) = E\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big] \quad \forall s \in S$$

L'équation de Bellman permet de calculer la fonction de valeur de façon récursive:

$$V_t(s) = \max_{a \in A}\Big[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s')\Big]$$

L'algorithme Value Iteration [66][67] (Algorithm 5) permet de trouver la politique optimale π^* qui maximise la fonction de valeur. Elle applique l'équation de Bellman à chaque itération. Pour un horizon infini, l'algorithme s'arrete quand l'améloriation de la valeur est inférieure à un seuil ϵ. La complexité est de $\mathcal{O}(|S^2||A|)$ pour chaque itération [68].

Data: S, A, T, R, H, ϵ
Result: optimal policy π^*
Assign V_0 arbitrarily $\forall s \in S$ $t \leftarrow 0$ **while** $\max_{s \in S}(|V_t(s) - V_{t-1}(s)|) < \epsilon and t < H$ **do**
  forall $s \in S$ **do**
    $$V_t(s) = \max_{a \in A}\Big[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s')\Big]$$
  end
  $t \leftarrow t + 1$
end
forall $s \in S$ **do**
  $$\pi^*(s) = \arg\max_{a \in A}\Big[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_t(s')\Big]$$
end
return π^*

Algorithm 5: Value Iteration

A.2.2.2 Processus Décisionnels de Markov Partiellement Observables

Les Processus Décisionnels de Markov Partiellement Observables sont une extension aux MDP pour les systèmes à état partiellement observable.

Définition *(POMDP)* A.1. Un POMDP est défini comme une tuple $< S, A, T, \Omega, R, O, H >$, où:

- S est un ensemble discret et fini d'états s;
- A est un ensemble discret et fini d'actions a;
- $T : S \times A \mapsto \Pi(S)$ est une fonction de transition probabiliste, telle que $T(s, a, s') = Pr(s'|s, a)$;
- Ω est un ensemble discret et fini d'observations o;
- $R : S \times S \times A \mapsto \Re$ est une fonction de récompense;
- $O : S \times S \times A \mapsto \Pi(\Omega)$ est une fonction d'observation probabiliste.
- H est l'horizon de planification;

Dans un POMDP, l'agent ne connait pas l'état réel s du système, mais il maintient une croyance distribuée sur S. A chaque instant t l'agent a une croyance b_t sur l'état du système:

$$b_t(s) = Pr(s_t = s)$$

Quand l'agent exécute une action a et obtient une observation o, il met à jour son état de croyance de la façon suivante:

$$\tilde{b}_o^a(s') = \frac{\sum_s O(s, a, s', o) T(s, a, s') b(s)}{\sum_{s'} \sum_s O(s, a, s', o) T(s, a, s') b(s)} \quad \forall s' \in S$$

Un POMDP est équivalent à un Belief-MDP (BMDP)[84], en considérant chaque état de croyance b comme si c'était un état dans un MDP.

***Définition** (BMDP)* A.1. Un Belief-MDP est un POMDP modélisé comme une tuple $< \mathcal{B}, A, t, r >$, où:

- $\mathcal{B}$ est un ensemble continu d'états b;
- A est un ensemble discret et fini d'actions a;
- $t(b, a, b')$ est une fonction de transition;
- $r(b, a)$ est une fonction de récompense.

avec:

$$t(b, a, b') = P(b'|b, a) = \sum_{o \in \Omega} P(o|b, a)\delta(b', \tilde{b}_o^a)$$

$$P(o|b, a) = \sum_{s \in S} \sum_{s' \in S} O(s, a, s', o)T(s, a, s')b(s)$$

$$\delta(x, y) = \begin{cases} 1, & \text{si } x = y \\ 0, & \text{autrement} \end{cases}$$

et

$$r(b, a) = \sum_{s \in S} R(s, a, s')b(s)$$

Du moment que l'espace des croyances $\mathcal{B}$ est continu, il n'est pas possible de résoudre le BMDP simplement avec Value Iteration. Cependant, la fonction de valeur des POMDPs à horizon fini a une caracteristique qui peut être exploitée : la fonction de valeur optimale V^* est linéaire par morceaux et convexe [83] (Figure A.2.2.2). Plus précisément, V^* est un ensemble d'hyperplans dans l'espace de croyance, qui représentent la fonction de valeur d'une politique possible, et qui sont définis par un vecteur de coefficients appelés $\alpha-$vecteurs.

La représentation par $\alpha-$vecteurs a été utilisé pour développer des algorithmes de résolution exacte de POMDP [88] [89] [90][86]. Ces algorithmes cependant nécéssitent un

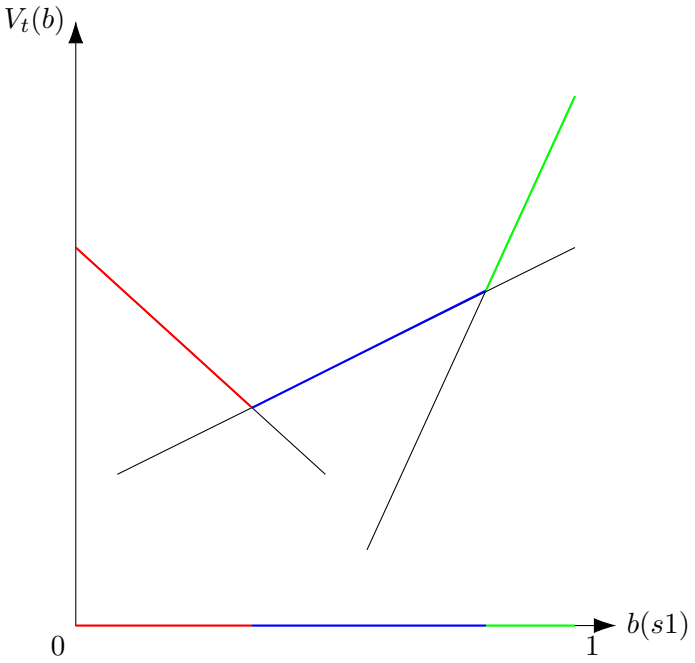


FIGURE A.2: Fonction de valeur optimale d'un POMDP

nombre exponontiel d'élements pour representer la fonction de valeur, et ne peuvent pas être utilisés pour des problèmes à grande échelle, comme la plupart des applications en réalité. Des algorithmes de resolution approximative ont été donc proposés par Lovejoy [91], Pineau et al. [92] et Spaan et al. [93]. Ces algorithmes n'utilisent qu'un ensemble fini $B \in \mathcal{B}$ de croyances ce qui réduit la complexité au détriment de l'optimalité.

A.3 Contributions

Dans cette Section nous présentons les principales contributions de cette thèse. D'abord, nous introduisons une approche théorique pour la génération de plans pour des activités jointes homme-robot qui puissent maintenir la coopération du partenaire humain tout au long de la tâche. Ensuite nous décrivons comment cette approche a été appliqué au scénario d'Escorte et implémentée dans l'architecture d'un robot réel. Nous présentons aussi les expériences menées pour évaluer l'approche et les résultats obtenus.

A.3.1 Planification de la coopération homme-robot

Le modèle décisionnel présenté dans cette thèse a été conçu pour adresser les problématiques suivantes:

1. Execution en temps réel
2. Incertitude sur le comportement humain
3. Environnement partiellement observable
4. Planification à horizon infini
5. Robuste

L'approche proposée consiste en un modèle à structure hiérarchique qui sépare l'aspect coopératif d'une activité jointe de la tâche en soi. Nous utilisons les POMDP pour planifier de manière efficace sous les contraintes 2 et 3. Le POMDP est construit en définissant les variables d'état, les actions et les observations du domaine d'application. Cependant, pour pouvoir tenir compte du niveau de coopération de l'humain il est nécessaire d'ajouter des éléments supplémentaires qui peuvent augmenter la complexité du POMDP au point d'en rendre la résolution intraitable. C'est donc pour en réduire la complexité que nous utilisons deux principes:

- *Abstraction d'état*: en effectuant une procédure d'abstraction sur l'espace d'état du POMDP, nous définissons des regroupements d'états qui sont ensuite utilisés comme états dans un POMDP abstrait. Ceci génère une structure hiérarchique où l'espace d'états des POMDP de niveau supérieur est plus réduit.
- *Décomposition en sous-tâches*: Nous considérons séparément les différents aspects d'une activité collaborative homme-robot. La *Tâche* est l'objectif qui doit être achevé indépendamment du niveau coopératif des agents. La *Coopération* est un état mental, appelé aussi Intention Jointe, où tous les agents sont engagés à mener à terme l'activité. La *Mission* est l'activité jointe où le robot et l'humain doivent coopérer pour achever leur but. La Mission est l'ensemble des parties Tâche et Coopération.

A.3.1.1 Structure Hiérarchique

La structure du modèle est présentée en Figure A.3. Elle consiste en trois niveaux, du niveau Primitif au niveau Status de Mission (le plus abstrait) en passant par le niveau Coopération au milieu.

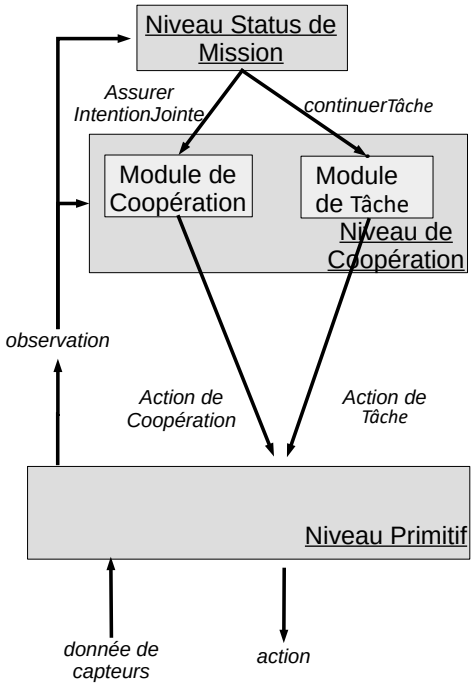


FIGURE A.3: Structure hierarchique

- Le niveau **Primitif** modélise le domaine d'application réel. Il fournit les observations des capteurs et exécute les actions de bas-niveau. Il agit comme interface entre le domaine d'application et les niveaux supérieurs, de façon à séparer la planification des problèmes d'exécution. Pour maintenir le modèle proposé le plus général et flexible possible, nous ne définissons pas le niveau Primitif de manière formelle. Cependant, il est représenté de la façon suivante: $< S^{bot}, A^{bot}, T^{bot}, \Omega^{bot} >$, où:
  - S^{bot} est l'espace d'états du domaine d'application;
  - A^{bot} est un ensemble d'actions primitives;
  - $T^{bot} : S^{bot} \times A^{bot} \mapsto \Pi(S^{bot})$ est une fonction de transition;
  - Ω^{bot} est un ensemble de observations;

  Les fonctions d'abstraction suivantes doivent être définies:
  - $S^{bot} \mapsto S^{mid}$
  - $A^{bot} \mapsto A^{mid}$
  - $\Omega^{bot} \mapsto \Omega^{mid}$

  où S^{mid}, A^{mid}, Ω^{mid} sont l'espace d'état, l'ensemble d'actions et d'observations du niveau Coopératif.

- Le niveau **Coopératif** est un niveau abstrait qui planifie la coopération homme-robot et le plan pour achever la tâche. Nous modélisons l'espace d'état S^{mid} de ce niveau de façon factorisé, comme le produit de variables d'état $x_1, ..., x_n$ telles que $S^{mid} = Dom(x_1) \times ... \times Dom(x_n)$. Nous définissons deux ensembles de variables: X^{task}, et X^{coop}. Les variables de coopération X^{coop} sont les variables introduites spécifiquement pour résoudre le problème de l'intention jointe. Elles modélisent les relations entre l'homme et le robot, et l'état mental de l'humain. Les variables de tâche X^{task} ne sont pertinentes qu'à l'aspect Tâche de la Mission et sont négligeables pour maintenir le niveau de coopération entre les agents. Il est donc possible de définir deux modules séparés au niveau Coopératif: le module Coopération et le module Tâche. Chaque module effectue la planification de manière indépendante en utilisant uniquement les variables d'état pertinentes, ce qui réduit la complexité du problème. Du moment que les modèles des deux modules sont indépendants, cette thèse ce concentrera sur le module Coopération du niveau Coopératif. Le module Coopération est modélisé comme un POMDP à horizon infini $< S^{co}, A^{co}, T^{co}, \Omega^{co}, R^{co}, O^{co} >$, où
  - S^{co} est l'espace d'états généré par X^{coop}
  - $A^{co} \subseteq A^{mid}$
  - $T^{co} : S^{co} \times A^{co} \mapsto \Pi(S^{co})$
  - $\Omega^{co} \subseteq \Omega^{mid}$
  - $O^{co} : S^{co} \times A^{co} \times S^{co} \mapsto \Pi(\Omega^{co})$
- Le niveau **Status de Mission** a le rôle de médiateur entre les modules de Tâche et de Coopération. Chaque module fournit une politique, et lors de l'exécution le niveau Status de Mission choisit s'il faut exécuter l'action de la Tâche ou l'action de Coopération. Ce choix dépend de l'état de coopération courante. A l'intérieur de l'espace d'état S^{co}, nous définissons un ensemble CS d'états coopératif dans lesquels l'intention jointe est assurée. Si l'état courant $s \in CS$, alors le niveau Status de Mission active le module de Tâche et le robot peut procéder avec la tâche à achever. Si $s \notin CS$, alors le robot doit d'abord ré-établir la coopération avec l'humain. Pour cela, le Status de Mission exécute l'action du module de Coopération.

A.3.1.2 Génération de politique

Nous allons nous concentrer sur la génération d'une politique à partir du POMDP du module de Coopération. Pour pouvoir facilement générer une politique à horizon infini qui puisse être implémentée comme une Machine à Etats Finis (FSM) sur un robot réel, l'approche adoptée est celle de représenter le POMDP comme un Belief-MDP Discrétisé (DBMDP), c'est à dire un Belief-MDP avec un ensemble discret et fini de croyances. Le DBMDP est ensuite résolu comme s'il s'agissait d'un MDP, avec Value Iteration. Pour pouvoir exécuter la politique obtenue comme un POMDP il est nécessaire de réintroduire les observations (manquantes dans un modèle MDP). Nous effectuons donc un passage d'une politique DBMDP à une politique POMDP. Ensuite, la politique est implémenté comme une FSM.

Pour éviter toute ambigüité, étant donné un Belief-MDP $< \mathcal{B}, A, t, r >$, nous définissons un Belief-MDP Discrétisé de la manière suivante :

Définition *(DBMDP)* A.1. Un BMDP Discrétisé est une tuple $< B, A, \tau, \rho >$, où:

- $B \subseteq \mathcal{B}$ est l'ensemble discret de croyances β;
- A est l'ensemble d'actions du BMDP;
- $\tau : B \times A \mapsto \Pi(B)$ est la fonction de transition entre croyances;
- $\rho(\beta, a)$ est la fonction de récompenses pour croyances, avec $\rho(\beta, a) = r(b, a)$.

Nous utilisons la notation

$$\beta(s) = Pr(x_1)Pr(x_2)...Pr(x_n)$$

de la meme manière que $b(s)$.

Pour pouvoir réintroduire les observations, une fois obtenue une politique $\tilde{\pi}$ en appliquant Value Iteration au DBMDP, nous définissons une fonction *belief shift* σ : $B \times A \times \Omega \mapsto B$ (FigureA.4). C'est une fonction de transition déterministe conditionnée par les observations. Elle est calculée de la manière suivante : étant donnée une croyance β, on calcule la mise à jour de la croyance avec l'équation A.2.2.2 et on obtient $\tilde{b}_o^a$, c'est à dire la croyance sur l'espace d'états S obtenue en observant o après avoir

effectué l'action a depuis la croyance $b = \beta$. Ensuite, cette croyance doit être discrétisée pour que le résultat appartienne à B :

$$\beta' = \arg\min_{\beta} \quad dist(\tilde{b}_o^a, \beta)$$

où $dist(\tilde{b}_o^a, \beta)$ est la distance entre $\tilde{b}_o^a$ et β, $\forall \beta \in B$.

Cette fonction nous permet de traduire une politique DBMDP $\tilde{\pi}$ en une politique POMDP π^σ qui associe à chaque croyance $\beta \in B$ une action a et un ensemble de croyances d'arrivée $SS^{\tilde{\pi}}$::

$$\pi^\sigma(\beta_i) =< a_i, SS_i^{\tilde{\pi}} >$$

avec $SS_i^{\tilde{\pi}} = \{\beta_i' = \sigma(\beta_i, \tilde{\pi}(\beta_i), o)\}$ et $|SS_i^{\tilde{\pi}}| = |\Omega|$.

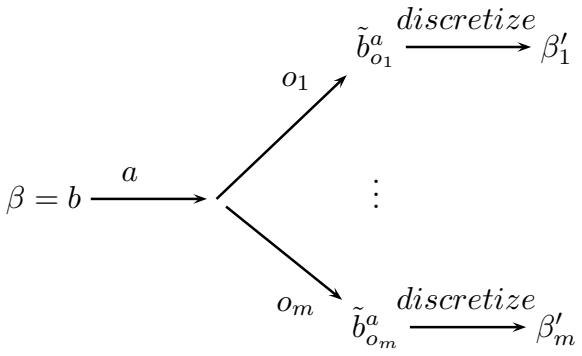


FIGURE A.4: La fonction belief shift

A.3.2 Implémentation du POMDP pour l'Escorte

A.3.2.1 La Mission Escorte

Dans le cadre du project COACHES, des robots de service, nommés Cadomus et Romus sont déployés dans un centre commercial à Caen pour accueillir et assister les visiteurs. En particulier, un des services que les robots peuvent fournir est la Mission d'Escorte, qui consiste à guider l'utilisateur vers une destination au choix.

Les robots sont dotés de caméras vidéo devant et derrière, de capteurs lasers, d'une tablette tactile avec une interface graphique, d'un micro et de haut-parleurs ainsi que d'un module de synthèse vocale.

Pour la planification de la Mission Escorte nous avons adopté l'approche présentée dans cette thèse. Le module de Tâche de la Mission, c'est à dire la Navigation vers la destination, est un module pré-existant dans l'architecture du robot. Nous avons donc implémenté le module de Coopération. Le POMDP de Coopération est modelisé de la façon suivante.

L'espace d'états est généré à partir de trois variables abstraites :

- Le *Niveau d'Attention* de l'humain: $Att = \{$ *Focalisé*, *Distrait*, *Perdu* $\}$.
- La *Distance de Proxémie* entre l'humain et le robot: $Dist = \{$ *Intime*, *Personelle*, *Sociale*, *Publique* $\}$.
- La *Position Relative* de l'humain par rapport au robot: $Pos = \{$ *Devant*, *Gauche*, *Droite*, *Derrière*

Dans l'espace d'état nous définissons l'ensemble d'états coopératifs CS.

Les actions définies pour le POMDP sont les suivantes:

- rester sur place
- avancer
- tourner à gauche
- tourner à droite
- attirer l'attention de l'humain
- naviguer

L'action *naviguer* consiste à activer le module de Navigation pour que le robot procède vers la destination. Le POMDP de Coopération joue donc aussi le rôle de Status de Mission layer: la fonction de récompense donne une forte récompense si l'action *naviguer* est exécutée dans un état $s \in CS$, de façon à assurer la coopération homme-robot avant de procéder avec la navigation.

Les caméras et les capteurs lasers permettent au robot de détecter et de suivre l'utilisateur. En particulier, un module de traitement de données fournit la position de l'humain, son

orientation et sa distance par rapport au robot. Ces informations sont utilisées comme observations pour estimer l'état du POMDP. A noter cependant que les capteurs ont des angles morts sur les cotés du robot. Les observations générées sont donc des combinaisons des informations suivantes:

- l'humain est détecté devant ou derrière le robot
- si détecté, le visage de l'humain est orienté ou pas vers le robot
- si détecté, la distance de proxemie à laquelle se trouve l'humain
- l'action courante effectuée par le robot

L'observation sur l'action courante permet au module de Coopération de savoir quelle action est en train d'exécuter le module de Navigation.

A.3.2.2 Implémentation et exécution

Une fois défini le modèle POMDP pour la Coopération de l'Escorte, celui-ci est discretisé en DBMDP. Le modèle DBMDP est résolu hors-ligne avec Value Iteration, ce qui fournit une politique $\tilde{\pi}$. Cette politique est ensuite traduite en une politique POMDP π^{σ} representée sous forme de FSM. Plus précisément, l'architecture du robot utilise des réseaux de Petri (PNP)[104] pour éxecuter les politiques de manière robuste. La traduction d'une politique POMDP sous forme de PNP est détaillée dans [103].

Au moment où la mission d'Escorte est initiée, le module de Navigation calcule en ligne un parcours vers la destination fournie par une base de connaissances. La navigation est cependant suspendue. Le plan de coopération est chargé par le module exécuteur de PNP. Un module de Évaluation de Situation traduit les donnés traitées par les capteurs en observations lisibles par le PNP, ce qui détermine les transitions d'états du réseau de Petri et l'estimation du niveau de coopération de l'humain. Quand l'exécuteur de PNP rencontre une action *naviguer* dans le plan, il procède selon le parcours calculé par le module de Navigation. Le plan de Coopération est toujours actif, car dès que le PNP ne se trouve plus dans un état coopératif, la navigation est à nouveau suspendue.

A.3.2.3 Escorte de groupe

L'approche présentée peut être adaptée pour guider un groupe de personnes au lieu d'une escorte individuelle. Ceci peut être fait de trois façons:

- Si le groupe inclut un leader, le robot guide le leader avec l'escorte individuelle.
- Si le groupe peut être considéré comme une seule entité, le module de Évaluation de Situation calcule la position et distance du barycentre du groupe, et la moyenne des niveaux d'attention, et fournit ses observations au plan de coopération individuelle.
- Le niveau Status de Mission de la hiérarchique peut être modifié pour estimer la coopération de l'ensemble du groupe. Au cas où la coopération ne serait pas suffisante, le Status de Mission devrait décider quel membre du groupe n'est pas en train de coopérer et donc activer le plan de coordination d'escorte individuelle avec cette personne.

A.3.3 Expériences et résultats

Nous avons effectué plusieurs expériences pour évaluer notre approche. Dans toutes les expériences, nous utilisons les mesures suivantes comme critères de performance :

- Le **Taux de coopération**, CR, mesuré comme le rapport entre le nombre de pas de temps où le système est réellement dans un état coopératif et le temps total d'exécution
- Le **Taux de navigation**, NR, mesuré comme le rapport entre le nombre de pas de temps où le robot exécute une action de navigation et le temps total d'exécution
- Le **Taux de navigation coopérative**, CNR, mesuré comme le rapport entre le nombre de pas de temps où le robot exécute une action de navigation dans un état coopératif réel et le temps total d'exécution
- La **Croyance coopérative**, BCS, mesuré comme le rapport entre la somme des croyances à chaque pas de temps que le robot a d'être dans un état coopératif et le temps total d'exécution

Toutes les expériences ont été effectuées plusieurs fois en variant le modèle de comportement de l'humain. Nous avons défini sept modèles de comportement, qui changent les probabilités que l'humain soit dans un état *Focalisé*, *Distrait* ou *Perdu* et qui représentent différents niveaux d'engagement à la tâche jointe. En ordre de plus engagé à moins engagé, les niveaux sont: *AF, AaF, MF, HT, ML, AaL,* et *AL.*

Nous avons effectué 30 simulations pour chaque modèle de comportement et calculé les moyennes des mesures de performance. Les simulations ont été effectuées dans un domaine simplifié : une grille discrète 15 x 15 sans obstacles. Nous avons ensuite comparé les résultas obtenus en utilisant notre approche avec ceux obtenus avec un modèle partiel (privé de l'observation sur l'action courante) et la politique obtenue avec un algorithme de résolution de POMDP par grille discrète (l'algorithme de Lovejoy [91]). La politique de Lovejoy est aussi privée d'observations sur l'action courante, par cause de incompatibilité avec le programme de résolution[1]. Le modèle partiel nous permet de évaluer l'impact de l'absence de ces observations en comparaison avec le modèle complet.

La Figure A.5 montre les résultats obtenus, en comparant la performance (CNR), le taux de navigation, le temps d'exécution et l'erreur de croyance (calculée comme $|CR-BCS|$). Le niveau de performance est proche de 1 dans le cas AF car il s'agit du cas idéal où l'humain est toujours focalisé et coopératif. L'approche proposée et l'algorithme de Lovejoy obtiennent les meilleurs résultats, dû à une croyance plus précise sur l'état courant (Figure A.5(d)). Le modèle partiel, privé des observations sur l'état courant, subit une perte d'information, ce qui comporte une réduction des actions de navigation (Figure A.5(b)) et donc une augmentation du temps nécéssaire pour guider l'humain à destination (Figure A.5(c)).

D'autres expériences ont été ensuite effectuées dans l'environnement de simulation Stage, plus proche d'une exécution réelle du robot.

Dans les simulations effectuées, il arrive que le robot perde de vue la personne en tournant ou en se déplaçant trop loin. La fréquence de cette perte dépend du modèle de comportement de l'humain. Malgré la baisse de performance conséquente, comparé aux expériences effectuées dans le domaine simplifié, le taux de succès de la tâche est de 100% dans la plupart des niveaux d'engagement.

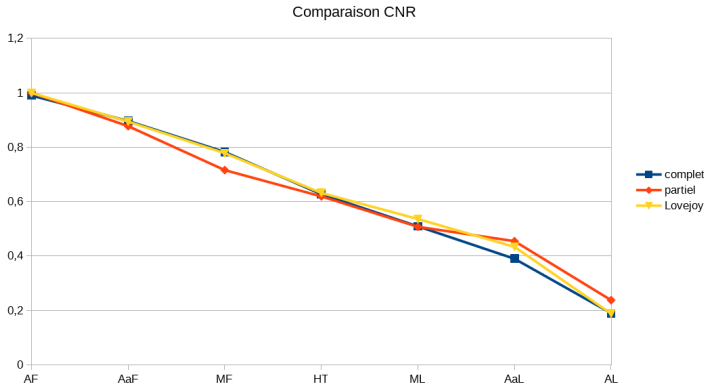
[1] `http://www.pomdp.org/code/index.html`

Enfin, nous avons implémenté avec succès la politique de Coopération dans un robot réel et effectué des démonstrations dans les couloirs de notre laboratoire. La vidéo de ces démonstrations est disponible au lien `https://youtu.be/r2ZizBcczGY`.

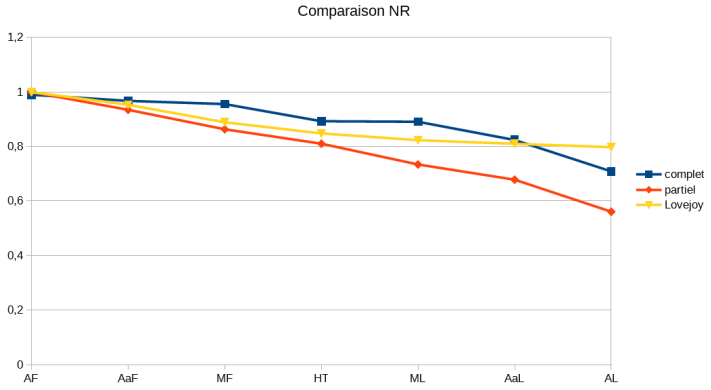
A.4 Conclusions

Nous avons formalisé une architecture décisionnelle pour planifier une tâche coopérative homme-robot capable de tenir compte du niveau d'attention de l'humain. Notre approche consiste à définir une structure hiérarchique, basée sur des POMDP, qui abstrait le domaine d'application pour se focaliser sur les relations entre agents. Elle sépare l'activité jointe homme-robot en deux aspects: la tâche à accomplir et la Coopération avec l'humain à assurer. Nous avons ensuite décrit comment cette approche peut être utilisée pour générer hors-ligne une politique POMDP à horizon infini, et comment l'implémenter dans un robot réel au sein d'une application pratique. Comme scénario d'application nous avons utilisé la mission d'Escorte dans le cadre du projet COACHES, qui consiste à guider un utilisateur vers une destination au choix dans un centre commercial. Nous avons effectué des tests et des simulations du scénario, en mesurant la robustesse de l'approche avec de nouveaux critères de performance et en la comparant avec un algorithme de l'état de l'art.

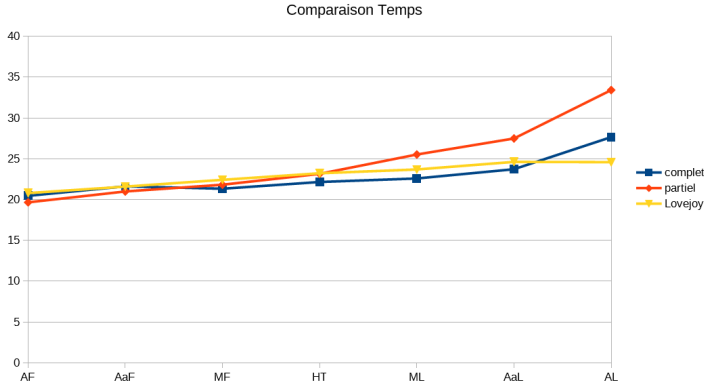
La thèse pourra être ultérieurement développée dans plusieurs directions. La tâche d'Escorte peut être généralisée au cas d'escorte de groupe. Nous pouvons effectuer des tests ultérieurs pour comparer notre approche avec d'autres algorithmes et avec un modèle unifié des deux aspects de l'activité jointe. Nous pouvons ajouter la possibilité de changer la vitesse du robot et la capacité de décider d'abandonner une tâche si l'engagement de l'humain est trop faible. Enfin, nous pouvons améliorer le modèle POMDP avec un apprentissage des fonctions de transition de l'humain.



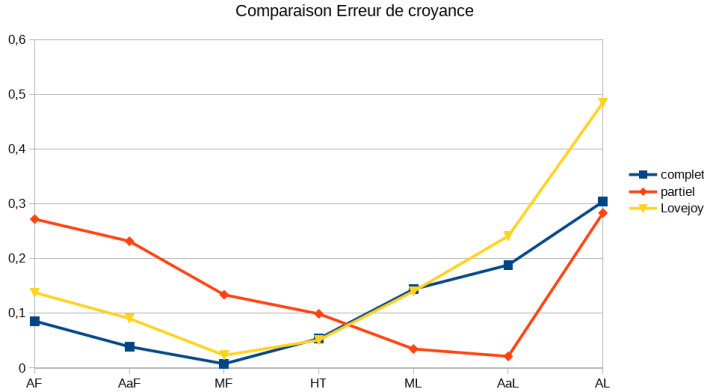
(a) CNR



(b) NR

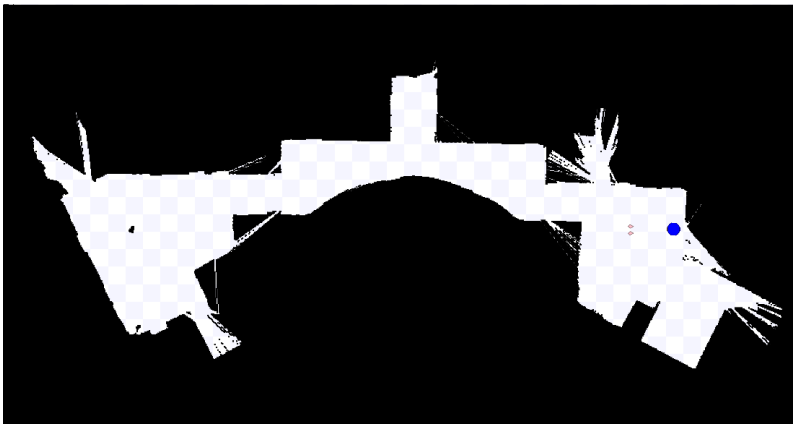


(c) Temps d'execution

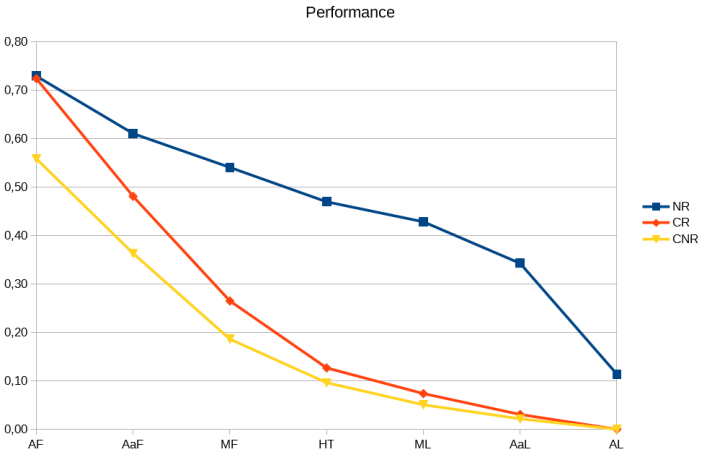


(d) Erreur de croyance

FIGURE A.5: Comparaison des résultats obtenus avec les trois différents politiques: modèle complet, modèle partiel, et algorithme de Lovejoy



(a) Le simulateur Stage



(b) Performance de simulation

FIGURE A.6: Résultats des expériences dans Stage

Bibliography

[1] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, et al. Minerva: A second-generation museum tour-guide robot. In *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, volume 3. IEEE, 1999.

[2] K. Pitsch, H. Kuzuoka, Y. Suzuki, L. Sussenbach, P. Luff, and C. Heath. The first five seconds: Contingent stepwise entry into an interaction as a means to secure sustained engagement in hri. In *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 985–991, Sept 2009.

[3] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[4] Balasubramaniyan Chandrasekaran and James M Conrad. Human-robot collaboration: A survey. In *SoutheastCon 2015*, pages 1–8. IEEE, 2015.

[5] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166, 2003.

[6] Stanislav Hristov Ivanov, Craig Webster, and Katerina Berezina. Adoption of robots and service automation by tourism and hospitality companies. 2017.

[7] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. The interactive museum tour-guide robot. 1998.

[8] Diego Rodriguez-Losada, Fernando Matia, Juan Manuel Lucas, Juan Manuel Montero, Miguel Hernando, and Ramon Galan. *Urbano, an interactive mobile tour-guide robot*. INTECH Open Access Publisher, 2008.

[9] Diana R Bueno, Eduardo Viruete, and L Montano. An autonomous tour guide robot in a next generation smart museum. In *5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI 2011)*. Citeseer, 2011.

[10] Aurélie Clodic, Sara Fleury, Rachid Alami, Raja Chatila, Gérard Bailly, Ludovic Brèthes, Maxime Cottret, Patrick Danès, Xavier Dollat, Frédéric Eliseï, Isabelle Ferrané, Matthieu Herrb, Guillaume Infantes, Christian Lemaire, Frédéric Lerasle, Jérôme Manhes, Patrick Marcoul, Paulo Menezes, and Université Paul Sabatier. Rackham: An interactive robot-guide. In *IEEE International Conference on Robot-Machine Interaction (RoMan)*, 2006.

[11] M Golam Rashed, Royta Suzuki, Antony Lam, Yoshinori Kobayashi, and Yoshinori Kuno. Toward museum guide robots proactively initiating interaction with humans. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, pages 1–2. ACM, 2015.

[12] Amit Kumar Pandey and Rachid Alami. Towards a sociable robot guide which respects and supports the human activity. In *CASE*, pages 262–267. IEEE, 2009.

[13] Illah R Nourbakhsh, Clayton Kunz, and Thomas Willeke. The mobot museum robot installations: A five year experiment. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 4, pages 3636–3641. IEEE, 2003.

[14] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro, and Norihiro Hagita. An affective guide robot in a shopping mall. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, HRI '09, pages 173–180, New York, NY, USA, 2009. ACM.

[15] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita. A communication robot in a shopping mall. *Robotics, IEEE Transactions on*, 26(5):897–913, Oct 2010.

[16] M. Shiomi, T. Kanda, D.F. Glas, S. Satake, H. Ishiguro, and N. Hagita. Field trial of networked social robots in a shopping mall. In *Intelligent Robots and Systems,*

2009. IROS 2009. IEEE/RSJ International Conference on, pages 2846–2853, Oct 2009.

[17] H.-M. Gross, H. Boehme, C. Schroeter, S. Mueller, A. Koenig, E. Einhorn, C. Martin, M. Merten, and A. Bley. Toomas: Interactive shopping guide robots in everyday use - final implementation and experiences from long-term field trials. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2005–2012, Oct 2009.

[18] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes: Challenges and results, 2003.

[19] Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*, pages 607–622. Springer, 2016.

[20] Edward T. Hall. The hidden dimension, 1966.

[21] M.P. Joosse, Ronald Walter Poppe, M. Lohse, and Vanessa Evers. *Cultural Differences in how an Engagement-Seeking Robot should Approach a Group of People*, pages 121–130. ACM, 8 2014. eemcs-eprint-24983.

[22] Phelipe AA Vasconcelos, Henrique NS Pereira, Douglas G Macharet, and Erickson R Nascimento. Socially acceptable robot navigation in the presence of humans. In *Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American*, pages 222–227. IEEE, 2015.

[23] Kheng Lee Koay, Emrah Akin Sisbot, Dag Sverre Syrdal, Mick L Walters, Kerstin Dautenhahn, and Rachid Alami. Exploratory study of a robot approaching a person in the context of handing over an object. In *AAAI spring symposium: multidisciplinary collaboration for socially assistive robotics*, pages 18–24, 2007.

[24] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[25] Kai Oliver Arras, Boris Lau, Slawomir Grzonka, Matthias Luber, Oscar Martinez Mozos, Daniel Meyer-Delius, and Wolfram Burgard. Range-based people detection and tracking for socially enabled service robots. *Towards Service Robots for Everyday Environments*, 76:235–280, 2012.

[26] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard. Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *2008 IEEE International Conference on Robotics and Automation*, pages 1710–1715, 2008.

[27] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *European Conference on Computer Vision*, pages 864–877. Springer, 2012.

[28] Omid Hosseini Jafari, Dennis Mitzel, and Bastian Leibe. Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5636–5643. IEEE, 2014.

[29] Stéphane Lallée, Katharina Hamann, Jasmin Steinwender, Felix Warneken, Uriel Martienz, Hector Barron-Gonzales, Ugo Pattacini, Ilaria Gori, Maxime Petit, Giorgio Metta, et al. Cooperative human robot interaction systems: Iv. communication of shared plans with naïve humans using gaze and speech. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 129–136. IEEE, 2013.

[30] Kerstin Dautenhahn, Bernard Ogden, and Tom Quick. From embodied to socially embedded agents – implications for interaction-aware robots. *Cognitive Systems Research*, 3(3):397 – 428, 2002. Situated and Embodied Cognition.

[31] Laëtitia Matignon, Abir Beatrice Karami, and Abdel-Illah Mouaddib. A model for verbal and non-verbal human-robot collaboration. In *2010 AAAI Fall Symposium Series*, 2010.

[32] B. Jensen, N. Tomatis, L. Mayor, A. Drygajlo, and R. Siegwart. Robots meet humans-interaction in public spaces. *Industrial Electronics, IEEE Transactions on*, 52(6):1530–1546, Dec 2005.

[33] Edgar A Martinez-Garcia, Ohya Akihisa, et al. Crowding and guiding groups of humans by teams of mobile robots. In *Advanced Robotics and its Social Impacts, 2005. IEEE Workshop on*, pages 91–96. IEEE, 2005.

[34] Bin Zhang et al. Adaptive motion control of social robot for guiding a users' group under dynamic environment.

[35] Michelangelo Fiore, Harmish Khambhaita, Grégoire Milliez, and Rachid Alami. An adaptive and proactive human-aware robot guide. In *International Conference on Social Robotics*, pages 194–203. Springer, 2015.

[36] Abir-Beatrice Karami. *Decisional Models of Human-Robot.* Theses, Université de Caen, December 2011. URL `https://hal.archives-ouvertes.fr/tel-01076430`.

[37] A. B. Karami, L. Jeanpierre, and A. I. Mouaddib. Partially observable markov decision process for managing robot collaboration with human. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 518–521, Nov 2009.

[38] Barbara J Grosz. Collaborative systems (aaai-94 presidential address). *AI magazine*, 17(2):67, 1996.

[39] Judith Bütepage and Danica Kragic. Human-robot collaboration: From psychology to social robotics. *arXiv preprint arXiv:1705.10146*, 2017.

[40] Cohen and Levesque. Teamwork. *Nous, Special Issue on Cognitive Science and AI*, 1991.

[41] Sanjeev Kumar, Marcus J. Huber, Philip R. Cohen, David, and R. Mcgee. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence*, 18:2002, 2002.

[42] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[43] David Morley and Karen Myers. The spark agent framework. In *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 714–721. IEEE, 2004.

[44] Anand S Rao. Agentspeak (l): Bdi agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer, 1996.

[45] Michael P Georgeff and Amy L Lansky. Reactive reasoning and planning. In *AAAI*, volume 87, pages 677–682, 1987.

[46] Rachid Alami, Raja Chatila, Aurélie Clodic, Sara Fleury, and Matthieu Herrb Vincent Montreuil. Towards human-aware cognitive robotics. In *In Cogrob 2006, The 5th International Cognitive Robotics Workshop (The AAAI-06 Workshop on Cognitive Robotics*, 2006.

[47] Ranjit Nair and Milind Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *J. Artif. Intell. Res.(JAIR)*, 23:367–420, 2005.

[48] Amit Kumar Pandey and Rachid Alami. A framework towards a socially aware mobile robot motion in human-centered dynamic environment. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5855–5860. IEEE, 2010.

[49] Frank Hoeller, Dirk Schulz, Mark Moors, and Frank E Schneider. Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1260–1265. IEEE, 2007.

[50] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):14–29, 2016.

[51] Tingting Liu, Jiaole Wang, and Max Q-H Meng. Human robot cooperation based on human intention inference. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 350–355. IEEE, 2014.

[52] Catharine LR McGhan, Ali Nasir, and Ella M Atkins. Human intent prediction using markov decision processes. *Journal of Aerospace Information Systems*, 2015.

[53] Rainer Stiefelhagen, C Fugen, R Gieselmann, Hartwig Holzapfel, Kai Nickel, and Alex Waibel. Natural human-robot interaction using speech, head pose and gestures. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2422–2427. IEEE, 2004.

[54] Sebastian Lang, Marcus Kleinehagenbrock, Sascha Hohenner, Jannik Fritsch, Gernot A Fink, and Gerhard Sagerer. Providing the basis for human-robot-interaction: A multi-modal attention system for a mobile robot. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 28–35. ACM, 2003.

[55] E Akin Sisbot, Raquel Ros, and Rachid Alami. Situation assessment for human-robot interactive object manipulation. In *RO-MAN, 2011 IEEE*, pages 15–20. IEEE, 2011.

[56] Mohammed Moshiul Hoque, Tomomi Onuki, Yoshinori Kobayashi, and Yoshinori Kuno. Controlling human attention through robot's gaze behaviors. In *Human system interactions (hsi), 2011 4th international conference on*, pages 195–202. IEEE, 2011.

[57] Lars Kopp and Peter Gärdenfors. *Attention as a minimal criterion of intentionality in robots*. Lund University, 2001.

[58] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[59] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[60] Gerald Jay Sussman. The virtuous nature of bugs. In *Proceedings of the 1st Summer Conference on Artificial Intelligence and Simulation of Behaviour*, AISB'74, pages 224–237, Amsterdam, The Netherlands, The Netherlands, 1974. IOS Press. URL `http://dl.acm.org/citation.cfm?id=3015486.3015502`.

[61] Daniel S Weld. An introduction to least commitment planning. *AI magazine*, 15 (4):27, 1994.

[62] Ronald PA Petrick and Fahiem Bacchus. Pks: Knowledge-based planning with incomplete information and sensing. In *Proc. of ICAPS-04*, 2004.

[63] Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins, Anthony Barrett, Dave Christianson, et al. Pddl—the planning domain definition language. 1998.

[64] Martin Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

[65] A. A. Markov. Theory of algorithms. *Journal of Symbolic Logic*, 22(1):77–79, 1957.

[66] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[67] Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[68] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[69] R. A. Howard. Dynamic programming and markov processes. 1960.

[70] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.

[71] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. Solving very large weakly coupled markov decision processes. In *AAAI/IAAI*, pages 165–172, 1998.

[72] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

[73] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *IJCAI*, volume 2, page 3, 1995.

[74] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 220–229. Morgan Kaufmann Publishers Inc., 1998.

[75] Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1-2):219–283, 1997.

[76] Amy McGovern, Doina Precup, Balaraman Ravindran, Satinder Singh, and Richard S Sutton. Hierarchical optimal control of mdps. In *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, pages 186–191, 1998.

[77] Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757, 2005.

[78] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[79] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[80] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Policy-contingent abstraction for robust robot control. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 477–484. Morgan Kaufmann Publishers Inc., 2002.

[81] Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, May 1992.

[82] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[83] Edward Jay Sondik. The optimal control of partially observable markov processes. Technical report, DTIC Document, 1971.

[84] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101: 99–134, 1998.

[85] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[86] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 54–61. Morgan Kaufmann Publishers, 1997.

[87] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[88] Edward J Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

[89] Michael L Littman. The witness algorithm: Solving partially observable markov decision processes. *Brown University, Providence, RI*, 1994.

[90] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

[91] William S Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations research*, 39(1):162–175, 1991.

[92] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[93] Matthijs TJ Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research*, pages 195–220, 2005.

[94] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press, 2004.

[95] David A McAllester and Satinder Singh. Approximate planning for factored pomdps using belief state simplification. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 409–416. Morgan Kaufmann Publishers Inc., 1999.

[96] Eric A Hansen and Zhengzhu Feng. Dynamic programming for pomdps using a factored state representation. In *AIPS*, pages 130–139, 2000.

[97] Hyeong Seop Sim, Kee-Eung Kim, Jin Hyung Kim, Du-Seong Chang, and Myoung-Wan Koo. Symbolic heuristic search value iteration for factored pomdps. In *AAAI*, pages 1088–1093, 2008.

[98] Joelle Pineau, Nicholas Roy, and Sebastian Thrun. A hierarchical approach to pomdp planning and execution. In *Workshop on hierarchy and memory in reinforcement learning (ICML)*, volume 65, page 51, 2001.

[99] Georgios Theocharous and Sridhar Mahadevan. Approximate planning with hierarchical partially observable markov decision process models for robot navigation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1347–1352. IEEE, 2002.

[100] Martijn Schut, Michael Wooldridge, and Simon Parsons. On partially observable mdps and bdi models. In *Foundations and Applications of Multi-Agent Systems*, pages 243–259. Springer, 2002.

[101] Luca Iocchi, Maria Teresa Lázaro, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Personalized short-term multi-modal interaction for social robots assisting users in shopping malls. In *International Conference on Social Robotics*, pages 264–274. Springer, 2015.

[102] Abdel-Illah Mouaddib, Shlomo Zilberstein, and Victor Danilchenko. New directions in modeling and control of progressive processing. In *ECAI*, volume 98, 1998.

[103] Luca Iocchi, Laurent Jeanpierre, Maria Teresa Lazaro, and Abdel-Illah Mouaddib. A practical framework for robust decision-theoretic planning and execution for service robots. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[104] Vittorio Amos Ziparo and Luca Iocchi. Petri net plans. Citeseer.

[105] Bram Bakker, Zoran Zivkovic, and Ben Kröse. Hierarchical dynamic programming for robot path planning. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2756–2761. IEEE, 2005.

Résumé

Objectif de cette thèse est le développent de méthodes de planification pour la résolution de tâches jointes homme-robot dans des espaces publiques. Dans les espaces publiques, les utilisateurs qui coopèrent avec le robot peuvent facilement se distraire et abandonner la tâche jointe. Cette thèse se focalise donc sur les défis posés par l'incertitude et imprévisibilité d'une coopération avec un humain. La thèse décrit l'état de l'art sur la coopération homme-robot dans la robotique de service, et sur les modèles de planification. Elle présente ensuite une nouvelle approche théorique, basée sur les processus décisionnels de Markov partiellement observables, qui permet de garantir la coopération de l'humain tout au long de la tâche, de façon flexible, robuste et rapide. La thèse introduit une structure hiérarchique qui sépare l'aspect coopératif d'une activité jointe de la tâche en soi. L'approche a été appliquée dans un scénario réel, un robot guide dans un centre commercial. La thèse présente les expériences effectuées pour mesurer la qualité de l'approche proposée, ainsi que les expériences avec le robot réel.

Mots-clés: interaction homme-robot, POMDP, activité jointe

Abstract

This thesis presents a novel method for ensuring cooperation between humans and robots in public spaces, under the constraint of human behavior uncertainty. The thesis introduces a hierarchical and flexible framework based on POMDPs. The framework partitions the overall joint activity into independent planning modules, each dealing with a specific aspect of the joint activity: either ensuring the human-robot cooperation, or proceeding with the task to achieve. The cooperation part can be solved independently from the task and executed as a finite state machine in order to contain online planning effort. In order to do so, we introduce a *belief shift* function and describe how to use it to transform a POMDP policy into an executable finite state machine. The developed framework has been implemented in a real application scenario as part of the COACHES project. The thesis describes the Escort mission used as testbed application and the details of implementation on the real robots. This scenario has as well been used to carry several experiments and to evaluate our contributions.

Keywords: HRI, POMDP, Joint Task