SOCIAL NAVIGATION OF AUTONOMOUS ROBOTS IN POPULATED ENVIRONMENTS

Noé Pérez Higueras

Social navigation of autonomous robots in populated environments

Escuela de Doctorado
Universidad Pablo de Olavide
Crta. Utrera, km. 1
41013 Seville, Spain

Social navigation of autonomous robots in populated environments

Tesis Doctoral

para la obtención del título de
Doctor por la Universidad Pablo de Olavide
Sevilla, Enero 2018

por

Noé Pérez-Higueras

Directores:
Fernando Caballero, Universidad Pablo de Olavide
Luis Merino, Universidad Pablo de Olavide

Tribunal

Prof. Dr. XXX Universidad XXX
Prof. Dr. XXX Universidad XXX
Prof. Dr. XXX Universidad XXX

Contents

Agradecimientos

Elaborar una tesis doctoral es un mérito personal debido al grado de compromiso, responsabilidad, disciplina, sacrificio e investigación que requiere (me autoagradezco el esfuerzo). Sin embargo, el autor, osea yo, no es, ni mucho menos, la única persona implicada con el producto final ya que se requiere de unos directores que guíen el proceso, eventuales patrocinadores para financiar el proyecto, el aporte de otros investigadores y un entorno familiar y social que permita desarrollar el potencial del doctorando. Por tanto, debo aprovechar la oportunidad de reconocer y agradecer la contribución de cada una de esas personas a este trabajo.

En primer lugar, y como no podría ser de otra manera, a mis padres. Tengo que agradecerles enormemente el gran esfuerzo que han realizado siempre para que me dedicara a estudiar y formarme y no me preocupara por nada más.

A mis directores, Luis Merino y Fernando Caballero (el dúo dinámico, nervio y calma, el Ying y el Yang, y tantas cosas más que se podría decir de ellos). Nunca podré terminar de agradecer la increíble oportunidad que me dieron de aprender de y con ellos y participar en proyectos tan increíbles. A Luis, le agradezco por guiarme, por siempre saber que hacer, y por siempre mirar por mi beneficio y crecimiento. A Fernando, por siempre estar de buen humor, por dar ese punto de ánimo y apoyo que muchas veces se necesita, y por remangarse y bajar al fango cada vez que se le necesita.

A la gente del grupo de Automática, Robótica y Mecatrónica de la Universidad de Almería. En especial a Francisco Rodríguez, que fue mi director de Proyecto Fin de Carrera y la persona que me dio la oportunidad de entrar en el mundo de la robótica y la investigación y que me sigue tratando tan bien a pesar de los años transcurridos. En esta línea, no me puedo olvidar de mi gran gran amigo y compañero de fatigas en nuestros inicios con el robot Peoplebot, José Pérez. Porque siempre sigamos "conectando hardware en caliente".

A mis compañeros del Laboratorio de Robótica de la UPO. Empezando por Javi, que me ayudó enormemente a adaptarme y siempre estuvo dispuesto a

echar una mano (Javi, sabes que tienes mi voto). A Rafa, trabajador incansable que nunca se ha quejado de nada. Y gracias por su decisiva aportación (arriba Rajufagro!). A Ignacio, el que más ha sufrido mis momentos de tensión en los proyectos. No sé como me ha aguantado tanto y a la vez, hemos pasado tantos momentos divertidos. A Chur, por ser un placer el trabajar con él, por ser tan positivo y crear tan buen ambiente. Y finalmente, a los jóvenes Padawans, Macarena y Álvaro, buenos momentos dentro y fuera del Lab. Que tengáis mucha suerte.

A mis Upitos, Carmen, Gabo y Leticia, merecidamente doctores todos, y grandes amigos. Hemos compartido infinidad de comidas aceitosas, risas y cervecitas, no ni na!

Finalmente, a los buenos amigos que he hecho durante mis años en Sevilla: mi "hermana" Paola, Carlos, Rocío, Cristina, Rosana, y Navarrete y Zaida. Gracias a todos ellos por su apoyo.

Resumen

En la actualidad, son más y más los robots móviles que conviven con nosotros en nuestra vida diaria. Como resultado, el comportamiento de estos robots, que comparten espacio con los humanos en entornos dinámicos, es un tema de intensa investigación en el campo de la robótica. Los robots deben respetar las convenciones sociales, garantizar el confort de las personas de su alrededor, y expresar un comportamiento legible, de manera que los humanos puedan comprender las intenciones del robot. Por lo tanto, los robots deben desplazarse en la proximidad de humanos cumpliendo estas reglas "sociales", lo que es denominado, el campo de la robótica, como "*human-aware navigation*". Estos comportamientos sociales no son fáciles de plasmar en expresiones matemáticas que los describan. En consecuencia, los planificadores de movimiento para robots clásicos que usan restricciones pre-programadas y funciones diseadas a mano, pueden fallar a la hora de mostrar correctamente este comportamiento social. En definitiva, es más fácil el demostrar un comportamiento socialmente aceptable en navegacin que el intentar definirlo matemáticamente. Por lo tanto, intentar aprender estos comportamientos a partir de demostraciones parece una approximación más adecuada.

Esta tesis aspira a dotar a los robots móviles con capacidades sociales de navegación en espacios concurridos. El trabajo aquí realizado se centra en encarar el problema de navegación social desde un punto de vista de aprendizaje por demostración (conocido por sus siglas en inglés LfD). De este modo, se han explorado y desarrollado diferentes técnicas y algoritmos de este campo con el objetivo de transferir comportamientos de navegación social a un robot usando para ello demostraciones de expertos humanos realizando la tarea que se desea aprender.

Por lo tanto, las contribuciones de esta tesis se enmarcan en el campo del aprendizaje por demostración aplicado a tareas de navegación social de robots móviles. Primero, se emplea una técnica de LfD, *Inverse Reinforcement Learning* en particular, para el aprendizaje de una política social de planificación de

movimiento local. Después, se presenta un novedoso algoritmo de aprendizaje que combina conceptos de LfD y planificadores de caminos basados en sampleo. Finalmente, se investigan otras novedosas aproximaciones, las cuales combinan técnicas de LfD, como *Deep learning* entre otras, y planificadores de caminos, para afrontar el problema de la navegación social. Todos los métodos propuestos son comparados con aproximaciones del estado del arte, y evaluados en diferentes experimentos con los robots reales empleados en los proyectos europeos FROG y TERESA.

Abstract

Today, more and more mobile robots are coexisting with us in our daily lives. As a result, the behavior of robots that share space with humans in dynamic environments is a subject of intense investigation in robotics. Robots must respect human social conventions, guarantee the comfort of surrounding people, and maintain the legibility so that humans can understand the robot's intentions. Robots that move in humans' vicinity should navigate in a socially compliant way; this is called human-aware navigation. These social behaviors are not easy to frame in mathematical expressions. Consequently, motion planners with preprogrammed constraints and hard-coded functions can fail in acquiring proper behaviors related to human-awareness. All in all, it is easier to demonstrate socially acceptable behaviors than mathematically defining them. Therefore, learning these social behaviors from data seems a more principled approach.

This thesis aims at endowing mobile robots with new social skills for autonomous navigation in spaces populated with humans. This work makes use of learning from demonstration (LfD) approaches to solve the problem of human-aware navigation. Different techniques and algorithms are explored and developed to transfer social navigation behaviors to a robot by using demonstrations of human experts performing the proposed tasks.

The contributions of this thesis are in the field of Learning from Demonstration applied to human-aware navigation tasks. First, a LfD technique based on Inverse Reinforcement Learning (IRL) is employed to learn a policy for "social" local motion planning. Then, a novel learning algorithm combining LfD concepts and sampling-based path planners is presented. Finally, other novel approaches combining different LfD techniques, like deep learning among others, and path planners are investigated. The methods proposed are compared against state-of-the-art approaches and tested in various experiments with the real robots employed in the European projects FROG and TERESA.

List of Figures

List of Tables

Chapter 1

Introduction

"Roads? Where we're going we don't need roads."
Back to the Future, 1985

1.1 Motivation

To endow robots with the capability of moving autonomously in a scenario is a crucial field of research for the robotics community. A motion planner is in charge of generating a trajectory that connects the initial robot configuration to the goal configuration, and that can be traversed by the robot safely and without collisions. Thus, proper motion planning is an essential tool for the autonomy and automation of any mobile robot.

Mobile robotics in domestic and dynamic environments has gained a lot of attention in the last few years. This increasing interest is motivated by the numerous useful tasks that a robot could perform for humans in our daily lives. This is called service robotics, where, from museum or exhibition guides to health-care tasks or domestic assistance, the robots face an extraordinary challenge: they have to share space with humans.

To successfully perform such tasks, robots must guarantee the safety of surrounding humans besides respecting the "social" conventions and preserve their comfort while moving in the scene. This thesis is focused on **human-aware robot navigation**, which is the research field that tackles the problem of robot navigation in environments with humans (Kruse et al., 2013). An example of such environment is shown in Fig. 1.1, where the robotic guide of the European project FROG is dealing with the visitors in the Royal Alcazar of Seville, Spain.

The navigation of pedestrians in crowded environments seems to be natural and effortless. Humans can predict the intent of others pedestrians and also to reason about the reactions of others to their actions (Goffman, 1971). Thus, they jointly adapt their movements at an early stage of an encounter, allowing

Figure 1.1: Capture of the robotic guide of the European project FROG in the Royal Alcazar of Seville, Spain.

smooth navigation to their destinations.

In these shared environments, the computation of navigation paths should be not only safe but also socially acceptable. Inducing such socially normative navigation behaviors into service robots is a problem not solved yet. This implies some challenges in the field of human-aware robot navigation (Khambhaita, 2017):

- First, robots must sense the people and differentiate them from the rest of obstacles in the scenario. Accurate detection and tracking of people are required. The current systems fuse the data from different sensors to estimate the person's position, orientation, and velocity. Moreover, the ability to predict short-term trajectories and mid-term destinations would be desirable.

- Robots must act and comply with the social norms. A navigation scheme has to capture the most pertinent criteria for calculating the costs or constraints that lead the robot to behave socially. However, determining the valuable information involved in the tasks and the definition of the appropriate costs and constraints is a key challenge. So, the study of the human navigation behavior is required to acquire this knowledge. The work developed in this thesis mainly addresses this issue.

- Finally, robots must react quick regarding the continuous changes of the dynamic scenarios and therefore, update their actions according to the new situations. Fulfilling these real-time constraints requires to re-compute the trajectories in a short time so that the robot behavior remains continuous and smooth.

In a nutshell, the general motivation of this thesis is to provide service robots with the capabilities that allow them to participate in such a natural and social navigation behavior.

1.2 Research Problem and Objectives

To show a socially normative behavior in navigation for mobile robots is an open field under intense investigation in the research community and even in the industry. Many different approaches from different points of view and different techniques have been applied to tackle the issue.

We face this research problem from the requirements of the European projects FROG[1] (Evers et al., 2014) and TERESA[2] (Shiarlis et al., 2015). First, an outdoor robotic guide for crowded, touristy places, like the zoo of Lisbon and the Royal Alcazar of Seville, was developed in the FROG project. This involved the navigation in very challenging scenarios with different kind of obstacles and terrains and very crowded situations. Later, a telepresence robot with autonomous capabilities for elderly was built in TERESA. In this case, a quick social navigation in home environments was necessary. Figure 1.2 shows captures of both robots navigating among humans. In both cases, the development of robot navigation systems with human-aware navigation skills was required.

The complexity of the human interactions that arise in navigation scenarios is high. It involves several factors that are difficult to quantify together with high variability in the execution of the tasks. Traditional methods for motion planning that compute time-optimal paths to the goal are not able to commit the different tasks of human-aware navigation. This problem was initially tackled by including simple pre-programmed costs and constraints into motion planners (Sisbot et al., 2007; Kirby et al., 2010). However hard-coded social behaviors can barely capture a general social navigation behavior. Moreover, in many implementations, like (Pacchierotti et al., 2006; Kirby et al., 2009), these costs are grounded in Proxemics theory (Hall, 1966). However Proxemics is focused on people interaction, and it could not be appropriate for navigating among people.

[1] `https://www.frogrobot.eu/`
[2] `http://teresaproject.eu/`

Figure 1.2: Captures of the service robots developed in the European projects FROG and TERESA. Left: FROG robot. Right: TERESA robot.

According to these observations, this thesis proposes to face the problem of human-aware robot navigation from the point of view of Learning from Demonstration (LfD) (Argall et al., 2009). This idea comes from the difficulty of determining the appropriate costs or constraints involved in social navigation and also the fact that pedestrians move in efficient ways in very complex situations with minimal effort. All in all, the socially compliant navigation behaviors are more natural to demonstrate than mathematically defining them. So, a set of examples of human experts performing the tasks can be used to try to capture the underlying behavior that the expert is demonstrating. Therefore, LfD seems an appropriate approach to learning these social behaviors and transfer them to a robot navigation system.

We make use of LfD methods with the aim of developing novel approaches applied to the problem of human-aware navigation of service robots. Thus, different LfD techniques are studied, extended and combined with other methods in this thesis to enhance the navigation systems with human-awareness capabilities. In particular, the approximations proposed to make use of techniques like Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000), Unsupervised Learning based on Gaussian Mixtures Models (GMMs) (Calinon, 2009) and Supervised Learning through Deep Networks (Goodfellow et al., 2016) in combination with sampling-based path planners.

The most of the proposed learning methods make use of demonstrations in static scenarios, in which dynamic features of the human movement are not taken into account or the pedestrians are considered still. Then, this learned behavior

is applied to real dynamic scenarios by re-planning at high frequency. However, the methods are general enough, and the possible modifications and extensions to learn from demonstrations with dynamic characteristics are discussed.

In summary, the complexity and variability of the human navigation interactions makes the problem difficult to define and describe mathematically. Thus, the primary objective of this thesis is to try to capture the underlying "rules" followed in navigation by observing examples of human navigation in crowded environments, and then, transfer them to robot navigation systems. Therefore, this thesis proposes a set of novel approaches based on Learning from Demonstration to tackle the problem of human-aware navigation for service robots.

1.3 Thesis Contributions

All in all, the main contribution is a set of novel methods that make use of different techniques of Learning from Demonstration, like IRL, GMMs, and FCNs, combined with different motion planners to endow service robots with socially-acceptable navigation behaviors. In particular, the main contributions can be summarized as follows:

- The application of the Inverse Reinforcement Learning technique to learn human-aware navigation behaviors and their integration into reactive motion planners for robot navigation.

- A novel combination of a Programming by Demonstration (PbD) technique, Gaussian Mixture Models (GMMs), with a sampling-based algorithm, Optimal Rapidly-Exploring Random Trees (RRT*), that is successfully applied to social navigation tasks like approaching a person or group or avoiding people.

- Development of a new Inverse Reinforcement Learning algorithm that learns the weights of a cost function employed by a sampling-based planner (RRT*) leading the planner to behave similarly to the expert demonstrations. Experiments of social navigation confirm the ability to learn these normative behaviors.

- A novel approach to learn human-aware path planning that makes use of Fully Convolutional Networks (FCN) to learn from expert's path demonstrations and allows to overcome the problem of manually designing (or identifying) the cost-map and relevant features for the task of robot navigation.

The contributions of this thesis are strongly linked to the work developed in the European projects FROG and TERESA, as commented previously. As a result, the methods developed in the thesis have been deployed in two robots in two realistic scenarios for several weeks. Most of the techniques and software developed is publicly available on the Github of the Service Robotics lab[3]. Also the datasets collected during the experiments in such environments are available for the research community[4]. Furthermore, the collaboration with other research entities in the mentioned projects gave rise to two research stays:

- Framed in the European FROG project, the deployment and testing of the navigation system with the real robot in real environments was developed as visiting researcher at the University of Twente, Netherlands, in collaboration with the Human Media Interaction group (HMI) under the supervision of Professor Vanessa Evers. The duration was one month and six days split into two stages in January 2014 and May 2015.

- As part of the European TERESA project, collaborative research on machine learning techniques was conducted as visiting PhD student in the research group of Intelligent Autonomous Systems (IAS), at the Informatics Institute of the University of Amsterdam, Netherlands, for 3 months (from January to April 2015), under the supervision of Dr. Simon Whiteson.

These contributions have led to the following publications in journals, conference proceedings, and workshop proceedings:

- (Ramón-Vigo et al., 2014). R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. Transferring human navigation behaviors into a robot local planner. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN*, 2014. doi: 10.1109/ROMAN.2014.6926347.

- (Pérez-Higueras et al., 2014). N. Pérez-Higueras, R. Ramón-Vigo, F. Caballero, and L. Merino. Robot local navigation with learned social cost functions. In *Proc. of the 11th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, volume 02, pages 618-625, 2014.

[3] `https://github.com/robotics-upo`

[4] FROG datasets: `http://robotics.upo.es/datasets/frog/upo/`
TERESA datasets: `http://robotics.upo.es/teresa/teresa-d43.html`

- (Shiarlis et al., 2015). K. Shiarlis, J. Messias, M. van Someren, S. Whiteson, J Kim, J Vroon, G. Englebienne, K. Truong, V. Evers, N. Pérez-Higueras, I. Pérez-Hurtado, R. Ramon-Vigo, F. Caballero, L. Merino, J. Shen, S. Petridis, M. Pantic, L. Hedman, M. Scherlund, R. Koster, and H. Michel. Teresa: A socially intelligent semi-autonomous telepresence system. In *Workshop on Machine Learning for Social Robotics at the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

- (Ramón-Vigo et al., 2016). R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. A framework for modelling local human-robot interactions based on unsupervised learning. In *International Conference on Social Robotics*, pages 32-41. Springer International Publishing, 2016.

- (Pérez-Higueras et al., 2016a). N. Pérez-Higueras, F. Caballero, and L. Merino. Learning robot navigation behaviors by demonstration using a RRT* planner. In *International Conference on Social Robotics*, pages 1-10. Springer International Publishing, 2016a.

- (Pérez-Higueras et al., 2016b). N. Pérez-Higueras, R. Ramón-Vigo, Ignacio Pérez-Hurtado, J. Capitán, F. Caballero, and L. Merino. A social navigation system in telepresence robots for elderly. In *Workshop Using Social Robots to Improve the Quality of Life in the Elderly at International Conference on Social Robotics*, 2016b.

- (Pérez-Higueras et al., 2017). N. Pérez-Higueras, F. Caballero, and L. Merino. Teaching Robot Navigation Behaviors to Optimal-RRT planners. *International Journal of Social Robotics*, Nov 2017. ISSN 1875-4805. doi: 10.1007/s12369-017-0448-1.

- (Pérez-Higueras et al., 2018). Noé Pérez-Higueras, Fernando Caballero and Luis Merino. Learning Human-Aware Path Planning with Fully Convolutional Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018. To appear.

1.4 Thesis Outline

The manuscript starts with a review of state of the art, in **Chapter 2**, about human-aware robot navigation, pointing out first the classical methods for robot navigation, and then the techniques and approaches in the literature employed to tackle the problem of human-awareness navigation.

Then, **Chapter 3** presents an approach based on Inverse Reinforcement Learning (IRL) to learn from expert demonstrations a local control policy for socially-acceptable robot motions.

Chapter 4 presents the European project FROG, in which an outdoors robotic guide for touristy, crowded places was developed. The method shown in the previous chapter is employed here to extend a state-of-the-art local motion planner for robot navigation by including a "social" cost term. The experiments performed with the robot in real controlled and uncontrolled scenarios are also presented.

In **Chapter 5**, an unsupervised learning approach based on Gaussian Mixture Models (GMMs) is employed to learn particular tasks of human-aware navigation, like approaching a person to interact with. Then, these Gaussian models are used to partially bias the sampling state space of a sampled-based path planner to generalize the behavior and overcome the possible obstacles in the scenarios.

Chapter 6 presents a novel learning algorithm that combines IRL concepts and sampling-based planners, to learn the weights of a cost function that leads the planner to behave similarly to the demonstrations provided.

In **Chapter 7**, the integration of the methods and techniques developed in Chapters 5 and 6, among others, in the TERESA European project are presented. They are applied to the navigation system of the real telepresence robot of the project and the results and evaluations obtained in different experiments are shown.

Chapter 8 proposes a novel learning method for human-aware path planning based on Fully Convolutional Neural Networks (FCNs). With this approach, we try to overcome some of the drawbacks encountered in the previous methods.

Finally, **Chapter 9** presents the conclusions and the open issues to be addressed in future work.

Chapter 2

State of the Art

"Wax on, wax off."
The Karate Kid, 1984

The focus of this thesis is robot navigation in spaces shared with humans. To navigate, a robot needs algorithms to plan a path from its current position to the goal in the scenario and compute the control commands that steer the robot to that goal. These algorithms must take into account the surrounding obstacles (static and dynamic) and, in case of human-aware navigation, the presence of people, to calculate a feasible, safe path to the goal without hitting any obstacle.

In this chapter, we first introduce the problem of the human-aware navigation for mobile robots. Then, we overview the classical motion planning methods to finally review the techniques employed in the literature to extend the motion planners to deal with the human-awareness issue.

2.1 Introduction to human-aware navigation

To endow robots with the capability of moving autonomously in a scenario is a crucial field of research in robotics. First, the robot has to be able to localize its configuration in the scenario and the configuration of the goal to reach; where we express a configuration as the position and orientation of a point the configuration space. Then, a motion planner is in charge of generating the trajectory that connects the initial robot configuration to the goal configuration, and that can be traversed by the robot safely and without collisions. So, proper motion planning is an essential tool for the autonomy and automation of any mobile robot. The localization problem is not addressed in this thesis so that we focus on the motion planning problem. Moreover, some human-aware navigation tasks may not require global localization, like following a person or approaching a person in the vicinity of the robot.

The traditional robot navigation systems try to compute paths to the goal regarding performance criteria as path length and time efficiency (Latombe, 1991b; Siciliano and Khatib, 2008). Therefore, they are may not be able to perform socially acceptable navigation in spaces shared with humans. It is clear that they have to treat people as agents different from obstacles and fulfill the requirements associated with the social norms. That means the robot must respect the social conventions and guarantee the safety and comfort of the surrounding people, besides showing an intelligible behavior. That is the basis of correct human-robot interaction in navigation (Kruse et al., 2013).

Many social robots for challenging applications in human environments have arisen from the research community in the last years. From the pioneers tour guides RHINO (Burgard et al., 1999) and its sister Minerva (Thrun et al., 1999) to the Fun Robotic Outdoor Guide FROG (Evers et al., 2014) or the passenger guide and assistant in airports SPENCER (Triebel et al., 2015). Or passing by the pedestrian assistant Obelix (Kummerle et al., 2015), the shopping assistant TOOMAS (Gross et al., 2009) or the museum guide Robotinho (Behnke et al., 2009). Also the extensive deployment of real mobile robots in shopping malls in Japan (Satake et al., 2013, 2015). Some captures of some of them can be seen in Fig 2.1.

Even nowadays, we can find different commercial service robots acting as hotel receptionist, robot bartender or client assistant among others. Some examples are the Softbank robot Pepper[1], the Sandbot robots[2], the Relay robot of Savioke[3] for hotels or the robotic bartender of MaccoRobotics[4].

All of them have faced the social challenges to a greater or lesser extent. Human-aware robot navigation is an intense field of research in the robotics community so that several publications can be found in the literature. In the following sections, we first present an overview of the different motion planning techniques that are relevant to this topic. Then, we focus on the evolution of these methods to include human-awareness skills in robot navigation. We emphasize the Machine Learning methods and techniques successfully applied to this problem and that are relevant to the work developed in this thesis.

2.2 Robot motion planning: an overview

Several methods and techniques have been successfully employed to generate the shortest path (or the quickest solution) without collisions from the initial robot

[1] `https://www.ald.softbankrobotics.com/en/robots/pepper`
[2] `http://en.sanbot.com/index.html`
[3] `http://www.savioke.com/`
[4] `https://www.maccorobotics.com/1`

(a) Rhino robot.

(b) Obelix robot.

(c) Toomas robot.

(d) Spencer robot.

Figure 2.1: Captures of some successful service robots arisen from research projects.

position to a goal position in configuration space.

In dynamic scenarios, the initial motion plan needs to be frequently updated to take into account the changes in the environment. In these conditions, the planning process is sometimes divided into two stages: global path planning and local motion planning (also known as obstacle avoidance or reactive motion planning). In the first stage, a path to the long-term goal is calculated. Then, a short-term plan, updated at high frequency, is employed to follow the long-term path and deal with previously unknown dynamic obstacles. Finally, these short-term trajectories are transformed to motion control commands that are sent to the robot. The original long-term path can also be re-planned, usually with a lower frequency than the local plan because of computational cost. Thus, a two-layered architecture is employed in many robot navigation systems, as depicted in Fig. 2.2. A further review of different approaches and concept definitions can be found in (Kunchev et al., 2006; Gonzalez-Bautista et al., 2015).

In this section, we follow this general planning scheme, which is not the only one possible, to describe the most widely-used methods and algorithms employed robot motion planning. We emphasize the significant planners related to the work developed in this thesis, and we indicate the requirements of the different planners to be modified and employed in human-aware navigation.

2.2.1 Global path planning

The environment of the robot (a continuous configuration space) for planning a path to the final goal can be represented in different ways. According to this criteria, three general approaches can be considered: combinatorial planning, sampling-based planning, and potential fields, as explained down below.

2.2.1.1 Combinatorial planning and graph search

Combinatorial planning discretizes the configuration space by explicitly capturing the connectivity of the space into a graph. This graph can be represented as 2D grids, topological maps or other valid structures that capture the connectivity of the environment. Thus, the combinatorial planning techniques find a path to the goal by producing a roadmap, which is a graph in which each vertex is a valid point of the configuration space, and each edge is a collision-free path that connects the different points through space.

Initial path planners considered the environment as a polygonal world where only the position (no orientation) was taken into account in the configuration space, and therefore no "social" information was considered. In these conditions, techniques like Visibility graphs (Lozano-Pérez and Wesley, 1979), Voronoi Graphs (Bhattacharya and Gavrilova, 2007; Garrido et al., 2006) or Cell Decom-

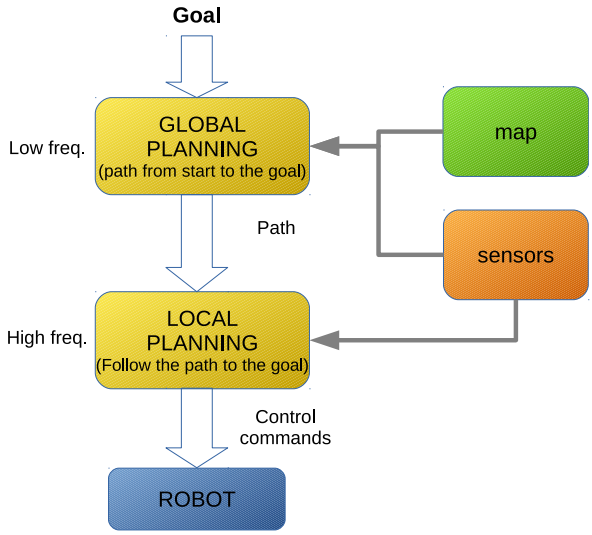


Figure 2.2: Two-layered navigation architecture.

position (Latombe, 1991a) were employed to decompose the space in simple regions (or cells) based on geometric characteristics and then, find the path following the adjacent cells from the initial cell to the goal cell.

More standard practice for path planning is to create a graph, in particular, a 2D occupancy grid, that separates traversable and non-traversable areas and use informed search algorithms to find the sequence of states or actions (path) that leads to the desired end state (goal).

The first search algorithm to find the shortest path in a graph was proposed by Dijkstra (Dijkstra, 1959). The idea was to expand vertices regarding to the so far unvisited neighbors iteratively. Later, the Dijkstra Algorithm was improved by using heuristics to estimate the cost from the current vertex to the goal, and thus, selecting in each step the vertex with the least estimated cost to the goal. This algorithm is the well-known A* algorithm (Hart et al., 1968), which is one

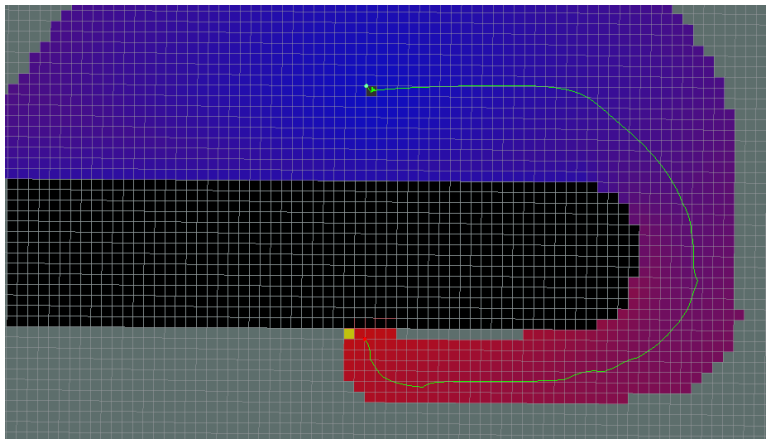


Figure 2.3: Example of path planning on a grid with the A* search algorithm implemented in ROS (Robot Operating System) (Quigley et al., 2009).

of most widely-used algorithms for robot path planning. An example can be seen in Fig. 2.3. The unvisited cells are shown in grey color, the obstacle cells in black and the visited cells present different colors regarding their costs (distance to the goal).

Several applications in mobile robotics have used A* as basis for improvement, such as the dynamic A* (D*) (Stentz, 1994), and D* Lite (Koenig and Likhachev, 2002), that incrementally repair the path keeping local modifications to avoid to replan the entire path in dynamic environments. A similar approach for efficiently replan is Field D* (Ferguson and Stentz, 2005) that also allows interpolation of costs of points not in the center of the grid cells. Another extension of A* is Theta* (Nash et al., 2007) where the vertices can have non-neighboring successors based on a line-of-sight test.

Graph search algorithms have some limitations. They require discretizing the configuration space so that the performance and time efficiency of the algorithm heavily depends on the granularity of the discretization. Moreover, the representation of the problem in the configuration space becomes infeasible when the complexity and dimensions increases, and an analytical solution may not be possible.

To deal with human-awareness, the graph search algorithms need to associate costs related to human presence in the space with the nodes of the graph. Moreover, the graph would require being frequently updated to account for the new dynamic situations, and these operations can increase even more the computational cost.

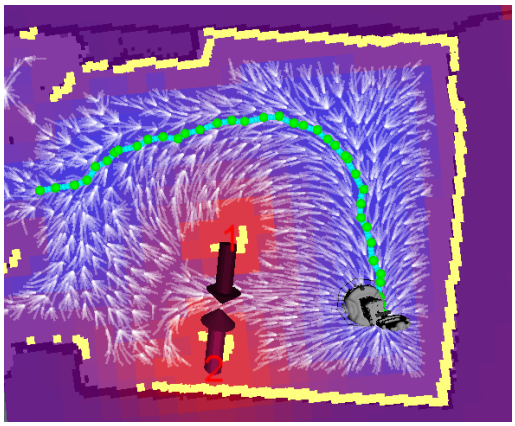


Figure 2.4: Example of real path planning with a RRT* algorithm employed in the TERESA telepresence robot. The tree is shown in white color and the resulting path in green color. The arrows represent the presence of two people in the scene.

2.2.1.2 Sampling-based planning

Sampling-based algorithms are powerful and popular planners that overcome some of the limitations of the graph search algorithms from a different point of view. Instead of characterizing the obstacles and free points of the configuration space and analytically finding collision-free connectivity, the configurations are randomly sampled from the configuration space and a collision-detection algorithm checks whether the sample is collision-free or not. Then, the sampled collision-free configuration is added as a node of a structure like a roadmap or a tree. A local planner is in charge of the connection between nodes in the structure.

These algorithms scale better than graph-search algorithms with large state spaces and high dimensionality. Moreover, the local planner employed to connect the nodes can take into account the kinematic and dynamic of the robotic platform, so these algorithms are easily extensible to handle these constraints. However, they offer weaker guarantees of optimality of the solution. Two families of sampling-based planners can be considered: the Probabilistic Road Maps and the Rapidly-Exploring Random Trees.

On the one hand, the Probabilistic Road Maps (PRMs) (Kavraki et al., 1996) are referred to as multi-query algorithms, as they first create a dense-enough roadmap and then answer queries by computing the shortest path that connects the initial node with the final node through the roadmap. On the other hand, the Rapidly-Exploring Random Trees (RRTs) (Lavalle, 1998; Lavalle et al., 2000)

are the single-query counterpart to PRMs. They make use of an incremental sampling to connect nodes in a tree structure that avoids the necessity to set the number of samples a priori. It also returns a solution as soon as the goal is connected to the tree, thus enabling on-line implementations. The RRT planner has been successfully employed in many robotic applications like (Cortés et al., 2007; Zucker et al., 2007; Kuwata et al., 2009), among many others.

Particular interest for robot path planning has the Optimal-RRT (RRT*) and Optimal-PRM (PRM*) algorithms (Karaman and Frazzoli, 2011). These algorithms are the asymptotically optimal and computationally efficient versions of their "standard" counterparts. These algorithms connect the nodes according to a defined cost function and continue refining the path after the initial solution has been found. Therefore, the behavior of the planner is determined according to the cost function employed, which is very interesting in problems like social navigation. Figure 2.4 shows an example of real path planning with a RRT* path planner using a cost function that takes into account the people in the scene (represented by arrows in the image), and thus, induces the path to keep some distance from the people.

Several variations of these algorithms have been presented in the literature. A combination of RRT and Theta* algorithms is presented in (Palmieri et al., 2016) for fast planning in dynamic environments. In the same context, the C-Forest algorithm (Otte and Correll, 2013) proposes to grow multiple search-trees in parallel; or the RRT^x algorithm (Otte and Frazzoli, 2015), which shows a sub-tree repairing method for RRT algorithms. In other sense, an approximation of RRT combined with stochastic optimization methods that use transition tests to accept or to reject new potential states is showed in (Jaillet et al., 2008; Devaurs et al., 2013). In (Gammell et al., 2014), a technique for reduction of the sampling space based on the calculation of a prolate hyperspheroid for problems that seeks to minimize the path length is described. Finally, a denominated Risk-RRT is presented in (Fulgenzi et al., 2010), where a RRT planner is augmented with the prediction of the short-term trajectory of dynamic obstacles coded trough Gaussian processes (Tay and Laugier, 2007).

To be employed in human-aware navigation, the sampling-based planners would require a way to bias the sampling process in favor of samples of the configuration space "interesting" for the social task. Moreover, in case of the planners RRT* and PRM*, The use of a user-defined cost function allows us to induce more complex behavior to the planner, which can be appropriate for human-aware navigation among other tasks. In this thesis, we mainly make use of a RRT* planner. We explore its capabilities for human-aware path planning combined with other techniques.

2.2.1.3 Potential fields

All the techniques discussed so far aim at capturing the connectivity of the configuration space into a graph. Potential Field techniques follow a different idea (Khatib, 1986; Koren and Borenstein, 1991). The robot, represented as a point the configuration space, is modeled as a particle under the influence of an artificial potential field. This potential field is composed by forces created by the environment, where the goal is an attractive force for the particle, and the obstacles are repulsive forces. The robot is then driven along the environment by the resultant force of the sum of the different forces involved.

Related to that, we mention the Navigation Functions (Koditschek and Rimon, 1990), which are a scalar-valued analytic map (similar to a potential field). They are constructed on a simplification of the workspace into a sphere-world (a compact connected subset of Euclidean n-space whose boundary is formed from the disjoint union of a finite number of $(n-1) - spheres$).

Some recent approaches are the work proposed by Arslan and Koditschek (2016a,b), where a reactive navigation vector field for a metric sphere world is constructed based on power diagrams (generalized Voronoi diagrams with additive weights (Aurenhammer, 1987)) to identify a collision-free convex neighborhood of a robot configuration. Another example is (Loizou, 2017), in which the workspace is transformed to a spherically bounded point world where a vector field can be efficiently computed.

In general, this method is fast and computationally efficient. However, the robot can get stuck in local minima, what makes it not recommendable for global motion planning. To use it in global motion tasks, to create a local-minima-free navigation function is necessary, and then perform gradient descent. Also, the configuration space needs to be explicitly represented what can be very costly. Therefore, this technique is adequate for local motion planning in general.

Finally, with the aim of adapting this technique to human-aware navigation, new forces related to the pedestrians in the vicinity of the robot could be added, as will be commented in Section 2.3.1.2.

2.2.2 Local motion planning

In highly dynamic environments, where the path planner does not takes into account the shape and the kinematic and dynamics of the robotic platform and a new path cannot be calculated at the required frequency for safe collision avoidance, the reactive motion planners (also known as local motion planners) become necessary.

These local planners are in charge of planning possible collision-free trajectories as fast as possible by accounting for the shape and kinematic constraints of

the vehicle. Several obstacle avoidance methods are proposed in the literature. The most extended approaches are summarized here.

- **Dynamic Window Approach (DWA)**. Probably one of the most used and well-known local planners is the DWA (Fox et al., 1997; Brock and Khatib, 1999), as it is implemented as a default local planner in the *move_base* framework for robot navigation of ROS (Robot Operating System)[5]. ROS is the set of software libraries and tools for building robot applications most used in the robotics community. DWA runs by choosing a velocity (linear and angular) from a set of velocities that maximizes certain objective function in a dynamic temporal window where the robot trajectory is simulated. The objective function accounts for distance to the goal, distance to the obstacles and distance to the path that is trying to follow. This way, DWA can generate smooth collision-free trajectories.

  Very similar to DWA is the Trajectory Rollout approach (Gerkey and Konolige, 2008), which is also implemented in the default navigation system of ROS. These algorithms only differ in how the robot's control space is sampled. Trajectory Rollout algorithm searches in the space of possible controls instead of searching the space of feasible trajectories. This means that DWA is a more efficient algorithm because it samples a smaller space, but maybe outperformed by Trajectory Rollout for robots with low acceleration limits because DWA does not forward simulate constant accelerations.

- **Potential fields**. Another approach employed as local motion planner is the potential fields, as commented in section 2.2.2. Some examples of robotic applications can be found in (Khatib, 1986; Brooks, 1986; Borenstein and Koren, 1991).

- **Elastic band**. In (Quinlan and Khatib, 1993; Khatib et al., 1997) a denominated "elastic band" approach is developed. The local planner computes an elastic band within a local area. The idea is to represent the path as a series of "bubbles" that feel repulsive forces from obstacles. This process in turn elastically modifies the robot path to balance the repulsive force of the obstacle and internal contraction force of the bubble-band. These bubbles are added or removed to keep the whole path fully covered by overlapping bubbles. Various heuristics are used to connect the center points of the bubbles and determine the distance to obstacles. A more recent approach (Rosmann et al., 2013, 2015) augments the elastic band with a series of time-difference values between each successive poses, instead of

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

a purely geometric path. This allows the band to be also deformed in time, that means to vary the robot velocity between two consecutive poses.

- **Velocity Obstacles algorithm (VO)**. Finally, a different approach is the family of the VO algorithms, originally presented in (Fiorini and Shiller, 1993, 1998). Instead of computing possible collisions based on positions as functions of time, they use velocity information. This consists of selecting avoidance maneuvers to avoid static and moving obstacles in the velocity space, based on the current positions and velocities of the robot and obstacles. Thus, the avoidance maneuvers are generated by selecting robot velocities outside of the Velocity Obstacles, which represent the set of robot velocities that would result in a collision with a given obstacle that moves at a given velocity, at some future time.

These local motion planners can also be extended/modified to include some "social" constraints. Some of the approaches proposed in the literature will be explained in the following section.

2.3 Human-aware navigation approaches

As previously commented, the classical motion planning methods need to be extended and combined with other techniques to achieve socially-acceptable robot navigation in spaces shared with humans.

Hereafter, we distinguish the work on human-aware navigation in the literature between different points of view. On the one hand, we summarize the approximations that manually design and code human-awareness behaviors into the robot navigation systems. On the other hand, we review the approximations that make use of techniques of Machine Learning, as Reinforcement Learning and Learning from Demonstration, to try to capture the socially normative behaviors and transfer them to the motion planners commented.

2.3.1 Hand-crafted social navigation

The objective of these approaches is to induce some socially-acceptable behavior by hard-coding functions into the motion planners. These functions are usually based on easily-measurable variables like distances to the near pedestrians, orientations, and velocities. This way, the robot is forced, for instance, to keep a predefined distance to pedestrians or to use a determined range of speeds in particular situations.

Several and different approaches to tackle this problem by hard-coding functions into the motion planners have been proposed in the literature. Here, we

try to classify some of these relevant approaches in groups that have used similar techniques or ideas to face this issue.

2.3.1.1 Proxemics

Many navigation systems include social components based on the Proxemics theory of Hall (1966). This approach classifies the distances that people keep between them in different personal areas of interaction depending on the relationship and intention.

Figure 2.5 shows an illustration of these areas. In general terms, the intimate distance (distance shorter than 0.45 m) is reserved for interactions like embracing, touching or whispering. The personal distance ($0.45 - 1.2$ m) is usually employed by friends. The social distance ($1.2 - 3.6$ m) is reserved for acquaintances and strangers. Finally, the public distance (> 3.6 m) is used for public speaking. Thus, the robots calculate their paths and movements based on the distances to near people, and, in general, trying not to invade the intimate spatial area of them.

Some examples of such social navigation systems based on Proxemics can be found in (Kirby et al., 2009; Barnaud et al., 2014; Mead and Matarić, 2016). Another example is (Truong and Ngo, 2016), where the formation of groups of people is also taken into account. A study of the formation of the different groups, called F-formations, in robotics applications can be found in (Cristani et al., 2011; Setti et al., 2015). In (Rios-Martinez et al., 2011; Spalanzani et al., 2012), the Risk-RRT approach is augmented with the concept of disturbance risk also based on Proxemics and F-formations of groups of people. However, the Proxemics theory is focused on people interaction, and it could not be suitable for navigation in crowded places, as shown in (Luber et al., 2012).

2.3.1.2 Social forces

The potential fields technique, as a reactive motion planner, have also been modified to include social skills. Helbing and Molnár (1995) employed this method to derive to the technique known as Social Forces Model (SFM). Besides the attractive force of the goal and the repulsive forces from the obstacles, the idea for social navigation is to include new repulsive forces between the robot and the people around based on distances and velocities. The aim is to obtain a resultant force that leads the robot to keep a comfortable distance to the people while still showing a desire to reach the goal.

An extended version of the social forces model is used in (Ferrer and Sanfeliu, 2014) for kinodynamic planning in urban environments. An illustration of the Extended Social Forces Models can be seen in Fig. 2.6, in which the differ-

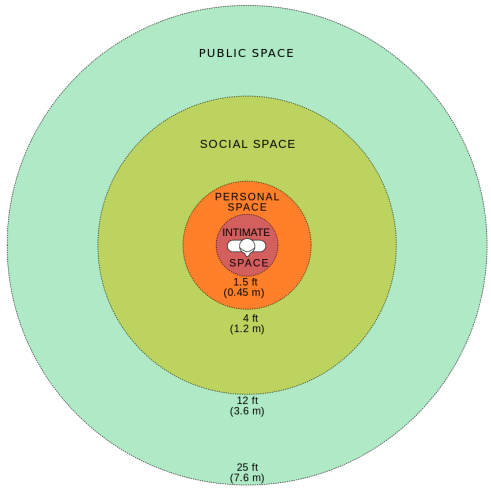


Figure 2.5: Illustration of the spaces for human interaction defined by Hall (1966). Image taken from Wikipedia.

ent repulsive and attractive forces are indicated with arrows. This model is also employed in (Ferrer et al., 2016, 2013) for the task of accompanying people walking side by side by including attractive forces to the person to be accompanied. Moreover, SFM has been successfully employed in simulators of highly-populated pedestrian environments too. However, the local forces are prone to lead to local minima and present difficulties to navigate narrow passages, as indicated in (Koren and Borenstein, 1991).

2.3.1.3 VO algorithms

Furthermore, some approaches have extended the Velocity Obstacles algorithm (VO) (Fiorini and Shiller, 1993), to deal with dynamic obstacles as pedestrians. In (Kluge and Prassler, 2004), a probabilistic variant of the VO algorithm (PVO) is developed to deal with uncertainty in the position predictions and measurements. Moreover, the concept of reflective navigation is presented as the idea that the mobile platform should reflect on the intentions of other dynamic agents to improve its navigation. Another extension, called Reciprocal Velocity Obstacles (RVO), is shown in (van den Berg et al., 2009). The main idea is to model the navigation as a collaborative task. There, the mutual evasion maneuvers of two agents are considered. So, both agents compute the RVO regarding the other agent producing collision-free motions. These techniques are mainly employed in multi-robot systems to avoid collisions. However, the RVO concept can be employed to model the behavior of groups of pedestrians. This latter is done in

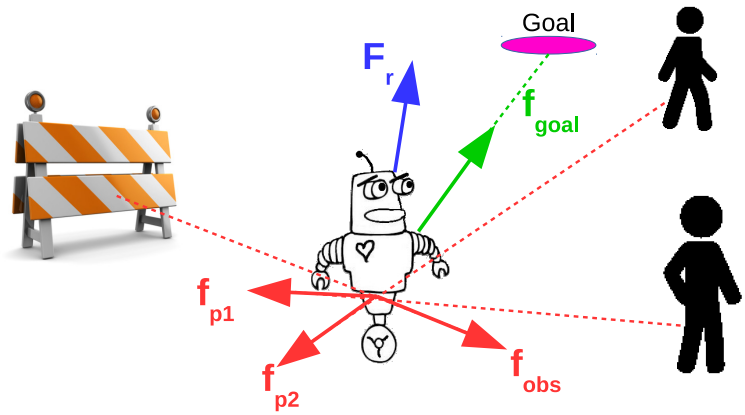


Figure 2.6: Illustration of the extended Social Forces Model. The repulsive forces are provoked by the people (f_{p1}, f_{p2}) and the obstacles (f_{obs}) are indicated with red arrows. the attractive force to the goal is showed with a green arrow (f_{goal}). The final resultant force is indicated with a blue arrow (F_r).

(Kim et al., 2015), in which an online method to predict pedestrian trajectories for improved human-robot interaction and collision-free navigation is presented. Ensemble Kalman Filtering is applied to estimate the parameters for the human motion model based on RVO in a crowded scene that optimally matches the data observed thus far. Then, using these RVO parameters adaptively computed, the crowd motion can be simulated to estimate the future trajectory of all individuals in the scene and estimate factors, such as their current goal.

2.3.1.4 Heuristics and other approaches

Other authors have manually derived functions for social behaviors as a collaborative task based on distances and approaching angles that have been hard-coded into motion planners. For instance, a heuristic for avoiding people is obtained in (Guzzi et al., 2013), or simple reasoning in crossing and overtaking situations is included in (Zanlungo et al., 2012). Another example is the work proposed by Khambhaita and Alami (2017a,b), in which a timed elastic band local controller is extended with social constraints based on Proxemics, time-to-collision and movement directions. The aim is to balance and tune the efforts between the human and the robot to solve a co-navigation task. In this sense, a probabilistic predictive model of cooperative collision avoidance and goal-oriented behavior is developed in (Trautman et al., 2015; Trautman, 2015) to tackle the "freezing robot" problem in very crowded environments.

A more sophisticated approach is the use of decision-making techniques. For

instance, a multi-policy Decision-Making approach is employed in (Mehta et al., 2016) to select among three pre-defined behaviors for social navigation. Another example that combines different techniques is (Ferrer and Sanfeliu, 2015), where a multi-objective cost-to-go function for a kinodynamic RRT planner is presented. This cost function also includes sub-costs related to social forces. Another different approach is followed in (Turnwald et al., 2016) where non-cooperative game theory is applied to formulate the problem of predicting the decisions of multiple humans that interact which each other during navigation and avoidance maneuvers.

Besides that, other works are based on observation of people moving in controlled experiments to extract conclusions that can be transferred into a robot. For instance, a crossing situation between two people is considered in (Pacchierotti et al., 2006) to perform a user study of passing distance in a corridor. A similar situation is evaluated in (Kruse et al., 2012) to extract the changes in the pedestrians' velocities to extend the HANP planner for social navigation employed in (Sisbot et al., 2007). Another interesting approach is presented in (Stein et al., 2013, 2014), where the human-aware navigation in very populated environments is accomplished by a probabilistic approach for selecting a human leader to follow, according to the robot's desired destination. In this way, the robot can take advantage of the humans' paths and behavior.

2.3.2 Reinforcement Learning

An interesting Machine Learning technique in robotics is Reinforcement Learning (RL), in which the problem or task is framed as a Markov Decision Process (MDP). Then, a policy (a map from the state of the system to the action to perform) is learned through trial-and-error interactions with its environment. Instead of explicitly detailing the solution to a problem, in RL the designer of a control task provides feedback regarding a scalar objective function (reward function) that measures the one-step performance of the robot. A complete review of RL techniques can be found in (Kober et al., 2013).

RL techniques has been extensively applied to robotic manipulation tasks but not very employed in robotic navigation. In (Hester et al., 2010), RL is employed with humanoid robots for the task of soccer penalty kick. A navigation task with obstacle avoidance is considered in (Smart, 2002); or the control of autonomous vehicle is tackled in (Hester et al., 2011). Basic navigation tasks combined with neural networks are learnt in (Thrun, 1995) and (Duan et al., 2008). More recently, a deep reinforcement learning approach for human-aware collision avoidance is presented in (Chen et al., 2017b,a), where a value function represented by a deep network is learned to implicitly encode cooperative and socially aware behaviors in a RL framework.

The RL framework presents some drawbacks when is applied to the human-aware navigation task. First, the use of discrete MDPs shows the difficulty of solving the MDP at each learning step in large state spaces and high dimensionality, which leads to high computational complexity. This limits the applicability of most of these approaches to small states spaces. Secondly and more relevant, the social navigation behavior is too general and complicated, so a reward function for such task is not easy to determine. The manually-designed reward functions may only capture some simple navigation situations leading to not very complex and human-like navigation behavior. For that reasons, RL is more used in more concrete and well-defined tasks where the design of the reward function remains simple.

2.3.3 Learning from Demonstrations

All the previous methods and techniques described so far try to code some socially-compliant behaviors into manually-designed heuristics or cost (or reward) functions that can be added to the path planners or reactive planners to fulfill the human-aware navigation issue. However, human-aware navigation implies social rules difficult to quantify and many variations in the execution of them. Thus, the manual design of such cost functions becomes hard. All in all, it is easier to demonstrate socially acceptable behaviors than mathematically defining them.

A different approach to overcome these issues is Learning from Demonstration (LfD). Within LfD, the translation from world states to actions (this mapping is called policy), is learned from examples, or demonstrations, provided by an expert performing the task to be learned. These examples are defined as a set of sequences of state-action pairs and are employed to derive a policy that reproduces the demonstrated behavior (Argall et al., 2009). This approach is in contrast to other techniques like Reinforcement Learning, where the policy is learned from experience and exploration of the environment. Therefore, LfD seems to be an appropriate technique for tackling the human-aware navigation problem.

LfD encompasses different techniques and methods. A review of various general LfD methods can be found in (Billard et al., 2008; Calinon, 2009). Here we present some social navigation approaches based on LfD according to their classification in other families of machine learning techniques like supervised learning and unsupervised learning. We also pay special attention to a particular supervised-learning method very employed in robot navigation, such as Inverse Reinforcement Learning.

2.3.3.1 Unsupervised Learning

The goal of this family of machine learning methods is to infer a function to describe a hidden structure from "unlabeled" data. This means that the available observations for learning do not include any classification or categorization. Therefore, the accuracy of the output structure of the algorithms cannot be evaluated.

Prediction of the human motion is a useful insight that can be employed in robot social navigation. This issue has been addressed in an unsupervised learning fashion. Bennewitz (2004) and Bennewitz et al. (2005) describe how to learn a set of motion patterns for a given map of an environment from observations of human motion, using the Expectation-Maximization Algorithm.

Another example is (Luber et al., 2012), in which a set of dynamic motion prototypes is learned from observations of relative motion behavior of humans found in publicly available surveillance data sets. The learned motion prototypes are then used to compute dynamic cost maps for path planning using an any-angle A* algorithm. Also (Growing) Hidden Markov Models are employed by Vasquez et al. (2009) to learn motion patterns incrementally and, at the same time, it uses its current knowledge to predict human motion.

Other approximations are the LfD methods based on Gaussian Mixture Models (GMMs). GMMs have been successfully applied in robotic manipulators (Calinon et al., 2007). They are also employed in the Cross-Entropy method for motion planning proposed by Kobilarov (2011, 2012). There, GMMs are employed to encode multiple trajectories across multiple homotopies classes that reach the goal. Then, these learned density distributions are employed to bias the sampling process or a sampling-based planner algorithm to reduce the sampling space to interesting areas for reaching the optimal trajectory to the goal. This approach does not include social navigation components, though some social behavior could be induced if the trajectories employed to be encoded through GMMs were socially-acceptable paths. We explored this idea in (Ramón-Vigo et al., 2016), in which social navigation tasks are encoded through GMMs, as explained in Chapter 5.

2.3.3.2 Supervised Learning

Supervised learning is a general method for training a parametrized function approximator. This method requires training samples of input-output pairs from the function to be learned. The output of the algorithms is compared against the corresponding demonstrated output to iteratively refine the parameters of the function being approximated.

Inferring the pedestrians' movement has also been addressed from the point

of view of supervised learning. In several cases, Gaussian Processes (GPs) are employed for that (Rasmussen and Williams, 2006a). An example is (Wang et al., 2008) in which the pedestrians' movement is inferred from observed movements using Gaussian Processes. This method was extended for intention inference in human-robot interactions (Wang et al., 2013). This latter was not employed in human-aware navigation, but the method is general. Another example is (Trautman et al., 2015), where human demonstrations are employed to code into Gaussian Processes a cooperative navigation behavior to face the problem of robot navigation in very crowded environments.

Other techniques like neural networks have also been applied to tackle some human-aware navigation tasks. In (Shiarlis et al., 2017b), the authors employ a deep learning approach to directly infer a control policy for cloning social behaviors from demonstration. That is, the demonstrated actions are treated as labels for the corresponding observations, and a mapping from observations to actions is learned. They employ this method to learn two social behaviors: first, keeping a correct position and orientation while interacting with a group of people of dynamic size and position, and second, following two people as their relative positions change.

We also explore a supervised learning approach based on deep learning for human-aware path planning (Pérez-Higueras et al., 2018). This technique is presented in Chapter 8.

2.3.3.3 Inverse Reinforcement Learning

One successful technique applied to robot navigation tasks is Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000): the observations of an expert demonstrating the task are used to recover the reward (or cost) function the demonstrator was attempting to maximize (minimize). Then, the reward can be used to obtain a similar robot policy. An interesting characteristic of IRL techniques is that they try to learn the underlying behavior that the demonstrator is showing, and therefore, they allow to generalize to different scenarios or situations where to perform the learned task. Furthermore, the learned reward functions can be transferred to different planners.

On the one hand, most of the IRL approaches frame the problem as a Markov Decision Process (MDP), just like Reinforcement Learning approaches. So IRL also faces the same issues of MDPs: poor scalability with large state spaces (and typically continuous spaces) and high dimensionality. On the other hand, the IRL goal is to learn the reward function of the MDP. Thus, it frees the designer of manually tuning the parameters of this function, what is very pertinent for socially compliant navigation.

The IRL problem has been tackled from different points of view in the litera-

ture. The projection method proposed by Abbeel and Ng (2004) was followed by other well-known approximations. A probabilistic method based on the principle of maximum entropy is presented in (Ziebart et al., 2008), or a maximum margin structured prediction based on subgradients is employed in (Ratliff et al., 2006, 2009). In (Ramachandran and Amir, 2007), a Bayesian perspective is taken to solve the problem.

Furthermore, learning robot navigation tasks from an IRL perspective has been considered by several authors. In (Vasquez et al., 2014) an experimental comparison of different IRL approaches is presented. In (Henry et al., 2010), IRL is employed in a path planning task in crowded environment. In (Kim and Pineau, 2016), the densities and velocities of pedestrians observed via RGB-D sensors are used as features in a reward function learned from demonstrations via IRL techniques for local robot path planning. Also in (Herman et al., 2015), different levels of social acceptability of the navigation behaviors are learned by using IRL. Then, the proper level of social behavior is selected under constraints on expected time-limits and reliabilities.

Moreover, different solutions have been considered to deal with the main MDP drawbacks. The computational cost is managed in (Michini et al., 2013) by using a Bayesian nonparametric mixture model to divide the observations and obtain a group of simpler reward functions to learn. In (Levine and Koltun, 2012), deterministic MDPs with large, continuous state and action spaces are handled by considering only the shape of the learned reward function in the neighborhood of the experts demonstrations. Furthermore, the power of deep neural networks as function approximators is applied by Xia and Kamel (2016) to learn non-linear policies. Also, the well-known Maximum Entropy IRL (Ziebart et al., 2008) algorithm is extended by Wulfmeier et al. (2015) to use deep networks, and its application to path planning in urban environments is presented in Wulfmeier et al. (2016).

Others authors tried to use different representations instead of MDPs. In (Ratliff et al., 2006), the MDP can be replaced by an A* search algorithm. A graph-based approximation is employed in (Okal and Arras, 2016a) to represent the underlying MDP in a socially normative robot navigation behavior task. Another example is (Kuderer et al., 2015) where IRL is used to learn different driving styles for autonomous vehicles using time-continuous splines for trajectory representation. In (Kretzschmar et al., 2016) the cooperative navigation behavior of humans is learned regarding mixture distributions using Hamiltonian Markov chain Monte Carlo sampling.

In this thesis, the IRL technique is employed to learn socially compliant navigation behaviors that can be transferred to different motion planners. This method is explored and combined with other techniques as explained in Chapters

3 and 6.

Chapter 3

Transferring human navigation behaviors into a robot local planner

"You're gonna need a bigger boat."
Jaws, 1975

We face the general problem of endow mobile robots with socially-acceptable navigation behaviors in spaces shared with people. So, in this chapter, a first approach to Learning from Demonstration a "social" control policy for a mobile robot is presented.

The approach makes use of an Inverse Reinforcement Learning technique based on a discrete Markov Decision Process (MDP) to learn a local reactive planner that leads the robot to show a socially normative behavior.

Two publicly-available datasets of person motion in different scenarios are employed here to learn the cost functions. Furthermore, we propose different simple sets of features for capturing in the MDP models a correct socially compliant behavior. Finally, we analyze and compare these approaches with a Proxemics-based cost function and comment the limitations of the approach.

3.1 Introduction

Classical path planners will not solve the social navigation problem, as planners try to minimize time or length, which does not translate to social paths in general. This requires determining costs related to social compliance. Moreover, the benefits of human motion go beyond collision avoidance approaches. It provides cues about the intentions of the subjects that can be used to decide when and where to move. Learning and imitating motion behaviors of people can also bring benefits regarding navigation efficiency for the robot. Besides that, the robot's motion can become more predictable, improving acceptance by pedestrians (Kruse et al., 2012).

Thus, we take an approach of Learning from Demonstration (Argall et al., 2009), in which we can learn these costs and models from human motion data. This approach is appropriate in problems that are complex to describe mathematically, like human-aware navigation, and therefore are easier to demonstrate.

An example is (Kretzschmar et al., 2013), in which the behavior of pedestrians is learned from observed trajectories composed of observations of pedestrians and also trajectories obtained by teleoperating the robot. One way to implement learning from demonstrations successfully is through Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000). The observations of an expert demonstrating the task that we want to learn to perform, are used to recover what reward (or cost) function the demonstrator was attempting to maximize. Then, the reward can be used to obtain a corresponding robot policy.

Many different general IRL algorithms have been proposed in the literature by tackling the problem from different points of view, as we reviewed in Chapter 2. The initial projection method proposed by Abbeel and Ng (2004), the probabilistic method proposed by Ziebart et al. (2008) based on the principle of Maximum Entropy, or the approach presented in (Levine et al., 2011), in which Gaussian Processes are employed to represent reward function instead of a linear combination of a set of features.

An interesting approach applied to human-aware navigation is (Henry et al., 2010). There, the authors present a path planner where they employ IRL to learn typical social behaviors from exemplary human trajectories moving in crowd environments. However, while in (Henry et al., 2010) the costs are used to path plans, here we employ these techniques to learn a local reactive planner. This is a control policy that maps from states to robot motion commands (linear and angular velocities), thus providing direct control of the robot. This can be combined with other planning techniques at higher levels while alleviating the complexity associated with learning.

In this chapter, we present an IRL approach, in particular, the Gaussian Process Inverse Reinforcement Learning (GPIRL) approach (Levine et al., 2011), to learn a reactive motion planner that leads the robot to have a socially normative behavior. A thorough analysis of the learning procedure is described, as well as the data used for learning. Two datasets of person motion in different scenarios are employed here to learn the cost functions. We study the generalization of the obtained reward functions by comparing the motion behavior learned from one scenario when applied in the other one. We also explore if the combination of the two training sets improves the general behavior. Furthermore, we propose an MDP model with a simple set of features on which the reward function is depending on. Then, we augment the proposed model adding high-level features based on person densities in different regions around the robot. Finally, we ana-

lyze and compare these approaches with a Proxemics-based cost function (Hall, 1966).

3.2 Inverse Reinforcement Learning

The Inverse Reinforcement Learning problem deals with finding a reward function that can explain observed behavior. This function is represented, in many cases, as a weighted linear combination of terms or features defining the task. The motivation of this is twofold: in many problems (like human-aware navigation) it is hard to determine the relative weights of the terms involved in a correct reward function *a priori*. Moreover, the field is founded on the presupposition that the reward function, rather than the policy, is the most succinct, robust and transferable definition of the task (Abbeel and Ng, 2004). Hence, it seems likely that IRL may, in some domains, provide an effective form of apprenticeship learning (Ng and Russell, 2000).

In this section, we introduce the notation and definitions of the MDP-based IRL methods, and the particularities of the two IRL approaches employed here. First, the Maximum Entropy IRL approach, and second, the Gaussian Process IRL which is an extension of the former approach.

3.2.1 MDP-based IRL

IRL assumes that the behavior of expert in the environment from which we want to learn can be modeled by a Markov Decision Process (MDP). Formally, a (discrete) MDP is defined by the tuple $\langle S, A, T, R, D, \gamma \rangle$:

- The *state space* S is the finite set of possible states $s \in S$;
- The *action space* A is defined as the finite set of possible actions $a \in A$.
- the state transition function T, which indicates how the state evolves when executing action a, and that is modelled by the conditional probability function $T(s', a, s) = p(s'|a, s)$.
- $R(s, a)$, the reward obtained for executing action a at state s.

At every step, an action is taken and a reward is given (or cost is incurred). A function $a = \pi(s)$ that maps a state to an action is called a policy (or controller). To each policy, it can be associated a *value* V_π, the expected cumulative reward following that policy for D steps, $E[\sum_{t=0}^{D} \gamma^t R(s, a)|\pi]$. The value function can be expressed in recursive form as:

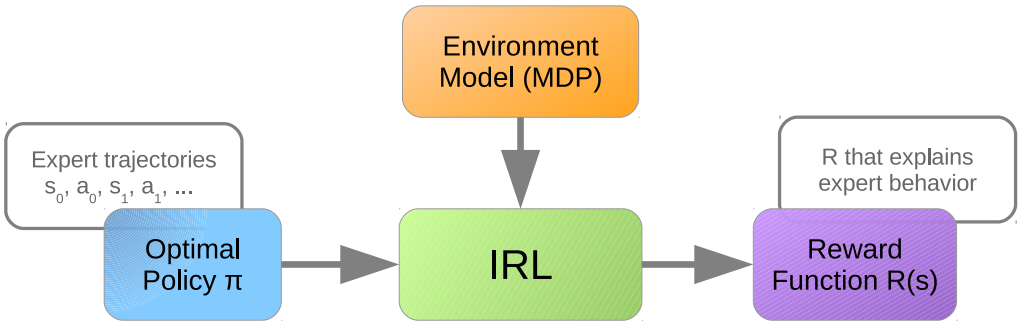


Figure 3.1: General diagram of Inverse Reinforcement Learning.

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_\pi(s') \tag{3.1}$$

Associated to the value function is also the $Q_\pi(s, a)$ function for each action a:

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_\pi(s') \tag{3.2}$$

To ensure that the sum is finite when $D \to \infty$, rewards are weighted by a discount factor $\gamma \in [0, 1)$

A policy π^* that maximizes the *value* V^* is called an *optimal* policy. The optimal value function is the fixed point of the recursion:

$$V^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V^*(s') \right] \tag{3.3}$$

and the optimal policy is thus:

$$\pi^* = \arg\max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_{\pi^*}(s') \right] \tag{3.4}$$

The optimal value function is also related to the optimal Q^* function as $V^*(s, a) = \max_a Q^*(s, a)$.

The objective of IRL is to determine the reward function $R(s, a)$ that the expert is following by observing the expert acting in the real world (demonstrations), and assuming that the expert is executing an (unknown) policy π according to the given MDP. In many cases, the reward function can be assumed to depend on a set of features $\theta(s)$, which are functions of the state. A general diagram of the IRL technique can be observed in Fig. 3.1.

Once the MDP model is defined, and given a set of examples in the form of expert demonstrations, $\mathcal{D} = \{\zeta_1, \cdots, \zeta_N\}$, where each demonstration consists of a path of state-action pairs $\zeta_i = \{s_{i,t}, a_{i,t}\}_{t=0}^{T}$, the IRL objective is to recover the unknown reward function $R(s, a)$.

3.2.2 Maximum Entropy Inverse Reinforcement Learning

This IRL approach uses a probabilistic model of the expert's behavior based on the principle of Maximum Entropy (Ziebart et al., 2008). The key consideration is that the demonstrated behavior is prone to noise and imperfect (human demonstrations are often suboptimal), so this uncertainty needs to be taken into account. This consideration fits perfectly in our problem of human-aware navigation, in which the set of human demonstrations are suboptimal.

Instead of assuming that the examples are sampled from the optimal policy π^*, this model considers that the probability of an expert path ζ_i (the sequence of states and actions visited) is proportional to the exponential of the differences of the rewards encountered along the path with respect to the optimally expected reward (the value V^*). Thus, maximizing the entropy of the distribution implies we maximize the likelihood of the observed data under the maximum entropy distribution (exponential family). Under this model, the paths with equivalent rewards have equal probabilities, and the paths with higher rewards are exponentially more preferred. Thus, given a reward function R, and denoting by Q^{*R} and V^{*R} the optimal Q and value functions for this particular reward R:

$$p(a_{i,t}|s_{i,t}, R) \propto \exp[Q^{*R}(s_{i,t}, a_{i,t}) - V^{*R}(s_{i,t})] \tag{3.5}$$

The probability for a episode is then:

$$p(\zeta_i|R) = \prod_{t=0}^{T} p(a_{i,t}|s_{i,t}, R) \tag{3.6}$$

and for the full demonstration

$$p(\mathcal{D}|R) = \prod_{i=1}^{N} \prod_{t=0}^{T} p(a_{i,t}|s_{i,t}, R) \tag{3.7}$$

Thus, the log-likelihood for the full demonstration is given by:

$$\log p(\mathcal{D}|R) = \sum_{i=1}^{N} \sum_{t=0}^{T} \left[Q^{*R}(s_{i,t}, a_{i,t}) - V^{*R}(s_{i,t})\right] \tag{3.8}$$

Gradient-based optimization can be applied to obtain the optima of the function. Thus, this log-likelihood can be differentiated to obtain the reward R that maximizes it.

This approach will also be employed as the basis in the development of the learning algorithm presented in Chapter 6.

3.2.3 Gaussian Process Inverse Reinforcement Learning

We consider the algorithm Gaussian Process IRL (GPIRL) (Levine et al., 2011) for solving the IRL problem. It employs the maximum entropy IRL model, described in the previous section, as the model for the demonstrations. And it extends that model by applying Gaussian Processes (GP) (Rasmussen and Williams, 2006b) to learn a non-linear reward function over the feature space $f(s)$.

Many IRL approaches represent the reward function R as a weighted linear combination of a set of provided features f describing the task, $R(s) = \omega^T f(s)$ (Abbeel and Ng, 2004; Ziebart et al., 2008; Ramachandran and Amir, 2007). In contrast, GPIRL makes use of Gaussian Process models to learn the reward as a highly non-linear function of features values that capture the demonstrated behavior.

A Gaussian process is a particular kind of statistical model where observations occur in a continuous domain, such as time or space and provides an alternative approach to regression problems. It is a non-parametric approach, in that it finds a distribution over the possible functions that are consistent with the observed data. As with all Bayesian methods it begins with a prior distribution and updates this as data points are observed, producing the posterior distribution over functions.

GPs are used to introduce some structure into the reward R, which can now be expressed as a non-linear function of features. The structure of this function is determined now by its kernel function; whose learned hyperparameters θ include the relevance of each feature besides capturing the structure of the reward.

The algorithm directly learns the output u of the induction points of the GP, which represent the rewards associated with feature coordinates X_u (these coordinates may simply be the feature values of all states or a subset of them). Then, the most likely values of u and the kernel hyperparameters θ are computed by maximizing their probability under the expert's demonstrations $\mathcal{D}$:

$$P(u, \theta|\mathcal{D}, X_u) \propto P(\mathcal{D}, u, \theta|X_u) = \tag{3.9}$$

$$\left[\int_R \underbrace{P(\mathcal{D}|R)}_{\text{IRL term}} \underbrace{P(R|u, \theta, X_u)}_{\text{GP posterior}} dR\right] \underbrace{P(u, \theta|X_u)}_{\text{GP probability}} \tag{3.10}$$

where the IRL term corresponds to the one expressed in (3.8), the GP posterior is the probability of a reward function under the current values of u and θ, and GP probability is the prior probability of a particular assignment to u and θ. Further details of the GP regression performed can be found at (Levine et al., 2011).

3.3 Learning a social cost function

This section describes the cost functions devised to model the navigation task. In particular, we are interested in learning a socially normative behavior in navigation in spaces shared with humans. This means that the robot should cross and avoid people in a socially-acceptable manner while navigating to its final goal in the space.

The most relevant aspect of the approach is to define the MDP model, and, in particular, the state and the features on which the reward function is depending on. This constitutes the central hypothesis considered here.

Small state space and a small set of features are considered here to keep the MDP solving process easily tractable. A first model based on features related to pairwise relative motions between two people is presented. Then, a second model considering more general aspects associated with the surrounding people is also shown.

3.3.1 Model 1

In principle, the actions of a person navigating among other people will depend on the state of all the persons close to the robot, plus other factors like obstacles and the person's goal. However, considering all the persons will lead to a large (and time-variant in size) state space. In (Henry et al., 2010), this is tackled by considering the density and flow direction as features, using them at the path planning level.

Here, the proposed model considers the generation of the velocity controls of the vehicle, and thus the action space corresponds to the linear and angular velocities of the robot $a = (v, \omega)$. Contrary to (Henry et al., 2010), we parametrize the state on the local robot/expert frame. This allows reducing the complexity

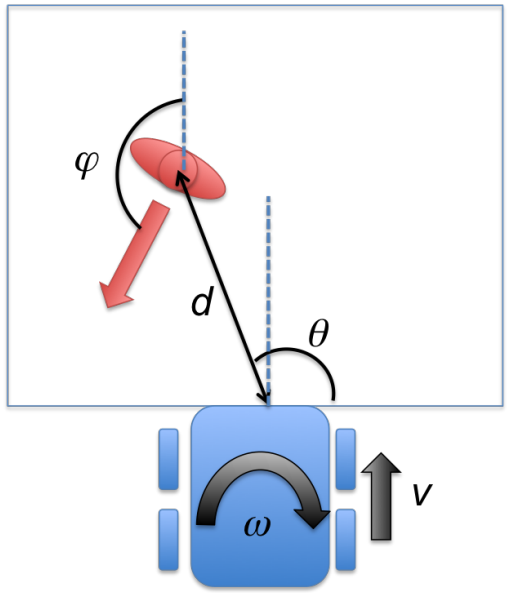


Figure 3.2: MDP model 1. The state is defined as the relative pose of the person with respect to the robot, encoded as the relative position of the person in polar coordinates (d, θ), and the approach angle φ. The actions (linear, v, and angular, ω, speeds) affect how this state evolves.

of the problem. Furthermore, in the model, we consider just pairwise relative motions between two persons (a robot and a person). The state is then defined by the relative position and orientation of the person with respect to the robot, encoded as $s = \begin{pmatrix} d & \theta & \varphi \end{pmatrix}^T$ (see Fig. 3.2). As the parameterization is local, the pose of the robot is not considered by the state.

The effects of the actions on the state are modeled by using simple kinematic equations, which are considered to be deterministic. Uncertainties are added to the person motion part, sampling several variations on the speed and angular velocity of the person and determining its future position. This way, the transition function $T(s', a, s)$ is determined.

One hypothesis that will be analyzed in this work is whether the model can be extended to cases with more persons utilizing the cost function learned applied to all the persons in the scene.

3.3.2 Model 2

The second MDP model is based on high-level features to define the reward function. In particular, the person densities in different regions in front of the

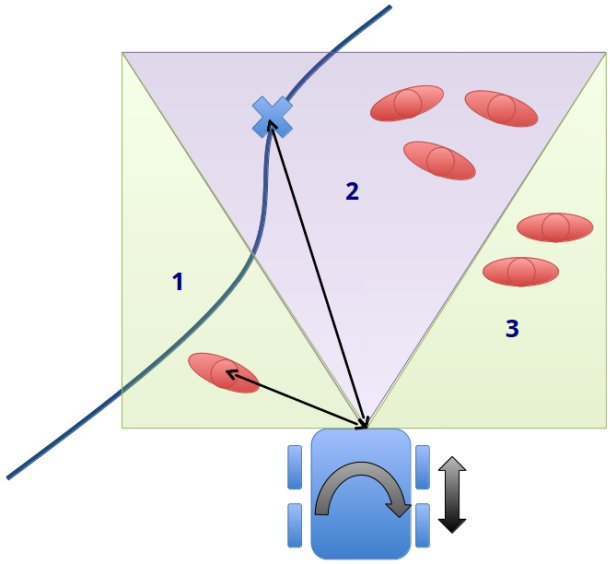


Figure 3.3: MDP model 2. The state is defined with the densities ($persons/m^2$) in the three regions in front of the robot.

robot are considered to parametrize the state on the robot frame. We use the same area as Model 1, but in this case, we divide it into three independent regions; one in front of the robot and two on the left and right sides (see Fig. 3.3).

Therefore, the state is encoded as $s = \begin{pmatrix}\rho_1 & \rho_2 & \rho_3\end{pmatrix}^T$, where ρ_i is the person density of the area i. Then, the transition function $T(s', a, s)$ is determined by considering how the densities in the regions are affected by the robot motion, and introducing at the same time uncertainties in the new density values due to the motion of people and the inflow of persons out of the field of view.

The development of this model aims at easily complementing the first model. The idea is to capture other navigation behaviors in crowded environments that the first model does not consider taking into account only the closest pedestrian. We alleviate the complexity of the problem and the computational cost by dividing the learning process into different reward functions. Thus, we do not add the new densities features to the previous model state to obtain a more significant and complete model. Instead, we develop a new model considering only the densities features and then obtaining the corresponding reward function. Then, we can use the reward function learned from model 1 and mix it with the new reward function obtained for the model of densities. The results will be shown in Section 3.6.

Figure 3.4: Example images of the BIWI Walking Pedestrian dataset used for learning (Pellegrini et al., 2009). Left: Sidewalk close to a hotel entrance. Right: entrance to the ETH main building.

3.4 Datasets for learning

As indicated above, it is hypothesized that learning the mentioned cost functions by observing how humans navigate among themselves will lead to socially normative behaviors.

As a source of examples on which the pedestrians motion is extracted, the public BIWI Walking Pedestrians dataset[1] (Pellegrini et al., 2009) has been used (see Fig. 3.4). The dataset consists of a set of detected targets of people walking in a two outdoors urban environments:

- The first proposed scenario (DS1) is a busy sidewalk next to a hotel entrance in Zurich (see Fig. 3.4, left).
- The second one (DS2), at the same dataset, is a bird view of the entrance to the ETH main building, in Zurich as well (see Fig. 3.4, right).

The global data covers about 25+ minutes of observation which has resulted in about 785 observed trajectories. Each of them consists of a set of positions and velocities of all persons and the corresponding timestamps, that are manually annotated at a rate of 0.4 seconds, which is a reasonable time step for the smoothness of the trajectories.

In the first scenario (sidewalk), the people are walking along the sidewalk crossing with people walking in the opposite direction, resulting in two input/output flows of pedestrians in both sides of the images. In this dataset, almost 64% of the captured frames contain at least one pedestrian. In more

[1] `http://www.vision.ee.ethz.ch/datasets/`

detail, there is 1 pedestrian in the 4% of the total frames, 2 pedestrians in the 5%, $3:8\%$, $4:11\%$, $5:8\%$, $6:5\%$, $7:6\%$, $8:3\%$, $9-18:14\%$ of the frames. By contrast, in the second dataset (ETH entrance), the people flow may appear from anywhere of the top and merge into the narrower entrance corridor, situated on the down part of the images. In this case, only about the 17% of the total captured frames contain at least 1 pedestrian ($1:2\%$, $2:3\%$, $3:1\%$, $4:1\%$, $5:3\%$, $6:1\%$, $7-11:6\%$).

3.4.1 Analysis of the data

A qualitative evaluation of the data provided in the datasets from the features employed in the two proposed MDP models is described here.

3.4.1.1 Features analysis of MDP model 1

Here we analyze the values of the features from the MDP model 1 for the data points of the datasets (DS1 and DS2). It should be recalled that all the features are computed locally to the expert.

Figure 3.5 shows the values obtained from the data set DS1. The bottom plot shows the polar coordinates of the closest person in the local frame. Several aspects can be highlighted. First of all, the most intimate person can be as close as 0.5 meters, well within the personal space according to Proxemics (Hall, 1966). It can also be seen that if the person is below 1 meter, it is typically located at the sides of the robot ($\theta \sim 0$ or $\theta \sim \pi$).

The upper-left and upper-right plots in the Fig. 3.5 show the distance vs. approach angle φ, and θ vs. φ respectively. As can be observed, the closest person is moving in the same direction ($\varphi \sim 0$ or $\varphi \sim 2\pi$), which indicates that they are walking in a group; while there are fewer cases in which the person cross in the opposite direction ($\varphi \sim \pi$). There are almost no examples in which persons cross with different angles, which indicates that persons try to follow the flow of people locally regarding direction. Also, it can be noticed how approaching persons ($\varphi \sim \pi$) are located at the sides of the robot.

Similar conclusions can be extracted from the training data set DS2, shown in Fig. 3.6.

3.4.1.2 Features analysis of MDP model 2

The densities in the three regions in front of the robot used in the MDP model 2 are analyzed here for the data points of the datasets DS1 (see Fig. 3.7) and DS2 (Fig. 3.8).

It is noteworthy that, in the DS1 case (sidewalk scenario), the density region 1 (left side of the person) is more populated than the other regions. This is

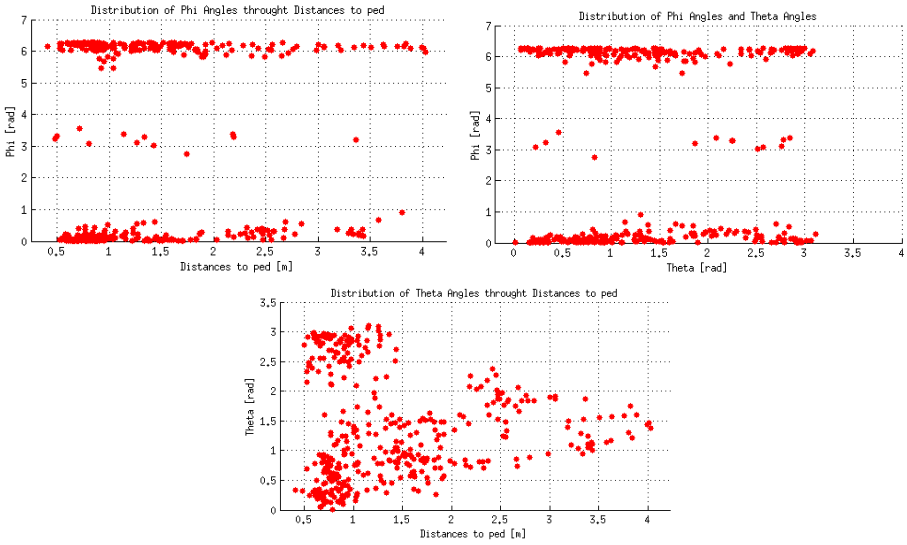


Figure 3.5: Annotated data of scenario DS1. Left: Approach angle φ vs. distance d in the local frame. Right: approach angle φ vs. θ. Bottom: polar coordinates (d, θ) of the closest person in the local frame.

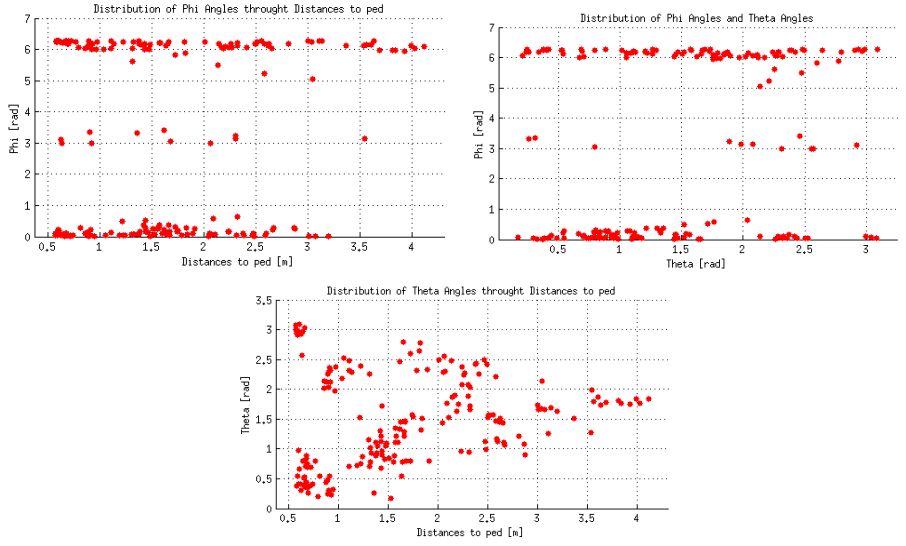


Figure 3.6: Annotated data of scenario DS2. Left: Approach angle φ vs. distance d in the local frame. Right: approach angle φ vs. θ. Bottom: polar coordinates (d, θ) of the closest person in the local frame.

because the pedestrians tend to form two flows on the sidewalk, where the people, in general, maintain some space on their left side for the pedestrians walking in the opposite direction.

Finally, it might seem that densities features are discretized, but this effect is due to the number of persons inside each area is always an integer, so the density evolves in steps of $1/area$. Notice that for density values in region 2, ρ_2, the x-axis takes from 0 to 0.5 in contrast with ρ_1 and ρ_3, that spreads from 0 to 1 per/m^2. That is because the central region of the model reaches double space (see Fig. 3.3). Thus, each consecutive horizontal edge to the right corresponds with an increment of one person, up to maximum six simultaneous for these datasets. It may occur that some of these edges are missing, so the next edge counts for an increment of two persons.

3.5 Description of the learning process

The details about the data used as expert demonstrations required by the GPIRL algorithm to learn the reward function and solve the related MDP are explained here.

3.5.1 Expert demonstrations

In our case, we employ the dataset to gather the examples from experts in the task of navigating among persons. Some persons are selected as "experts" among the pedestrians that are moving in the dataset. For each point in the trajectory followed by the person we extract:

- For Model 1, the state $s_i = \begin{pmatrix} d & \theta & \varphi \end{pmatrix}^T$ of the closest person within the local planning zone. This local environment (see Fig. 3.2) is defined as the region used for local planning on the robot, and it is defined as a rectangular region of 4x4 meters (4 meters in front and 2 meters at each side of the robot).

- For Model 2, the state $s_i = \begin{pmatrix} d & \theta & \varphi & \rho_1 & \rho_2 & \rho_3 \end{pmatrix}^T$ is completed with the values of person densities of the three regions considered (see Fig. 3.3). Here a rectangular region of 4x6 meters is defined (6 meters in front of the robot to gather more pedestrian information).

- The action performed by the expert at the same time step (for both models). In the particular implementation considered, the action space consists on the linear and angular velocities $a_i = \begin{pmatrix} v & \omega \end{pmatrix}^T$, in order to easily transfer them to the robot. The angular velocity ω is computed by measuring the change of orientation between consecutive poses of the expert.

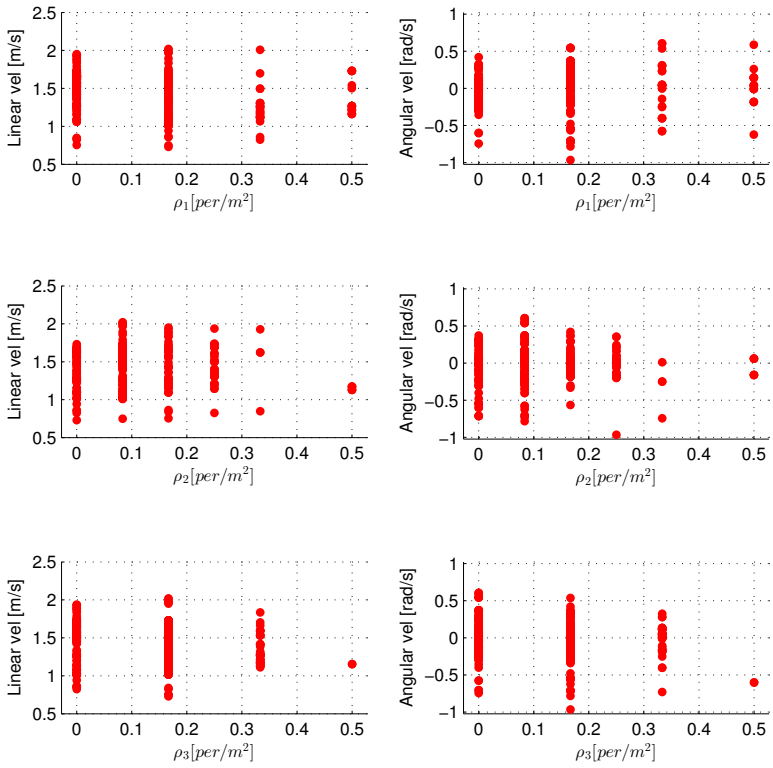


Figure 3.7: Density values in the three regions used in the MDP model 2 (ρ_1, ρ_2, ρ_3) for dataset DS1

When the closest person abandons the local planning region, the trajectory $\{s_i, a_i\}_{i=1}^{N}$ is stored as one episode for the training phase. A new episode is created for the next person. To have similar experiments, the number of samples is equalized for each episode, by dividing them in several if needed.

From each dataset, only moving pedestrians for which at least one person is within the local planner region for at least six time steps are selected as "experts". Furthermore, we impose that these pedestrians have to move at least 2 meters from their start point. Both conditions allow us to focus on interesting samples of pedestrians making social navigation. As a result, a training set per dataset is obtained. One of them is a set of 103 episodes from 51 different persons of a total of 420 tracked persons, and the other is a set of 47 episodes from 28 different

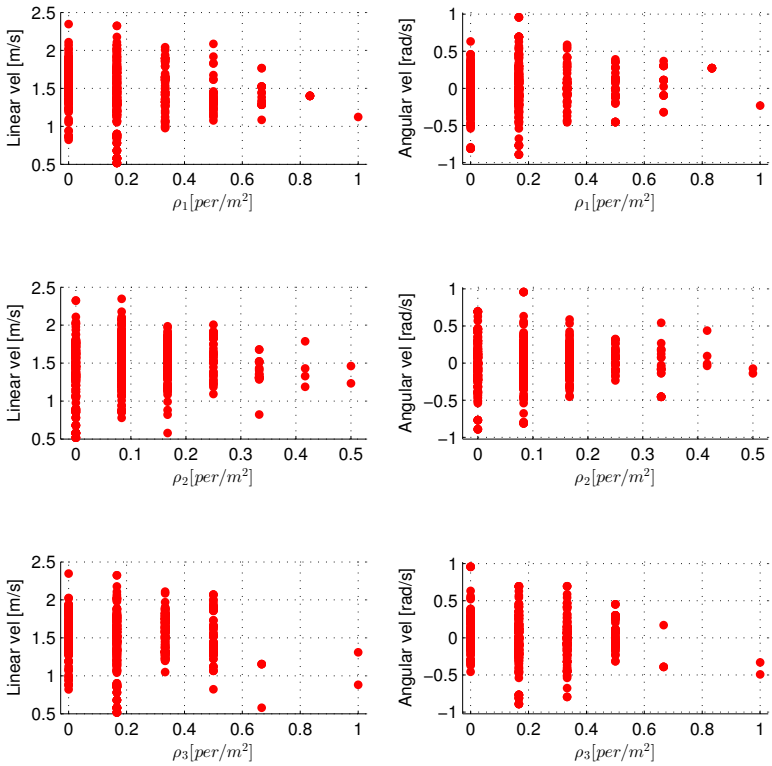


Figure 3.8: Density values in the three regions used in the MDP model 2 (ρ_1, ρ_2, ρ_3) for dataset DS2

persons of a total of 365 tracked persons. They are used to learn the reward function, and the rest of persons will be used in the evaluation to validate the estimated function.

3.5.2 Space discretization

The GPIRL algorithm uses a discrete MDP as a model. Therefore, the state and actions spaces need to be discretized. The local space used for the local planner is discretized as follows:

- The distance d is discretized into 11 bins of 0.5 m.

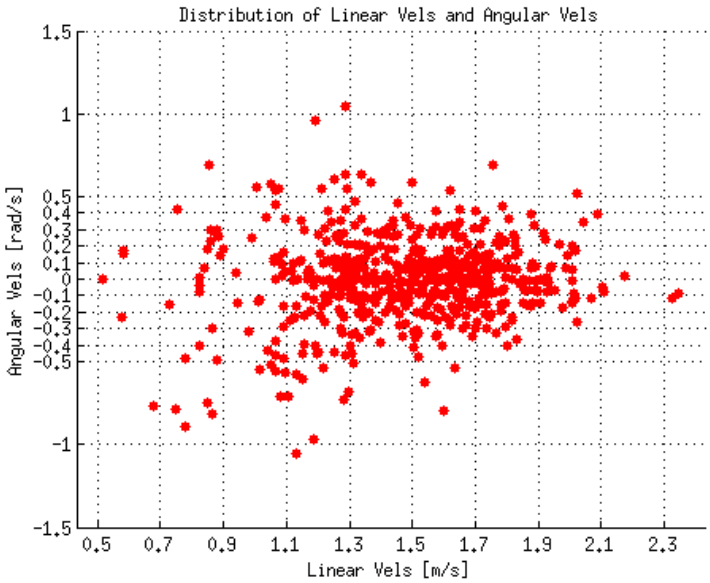


Figure 3.9: Actions (linear and angular velocities) of the training samples for both scenarios DS1 and DS2. The discretization bins employed are also shown.

- The relative angle $\theta \in [0\ \pi]$ is discretized into 6 bins of 0.62 rad.
- The person relative orientation $\varphi \in [0\ 2\pi)$ is discretized into 8 bins of 0.69 rad.
- The density value $\rho \in [0\ \infty)$ is discretized into 5 bins of 0.25 $persons/m^2$, starting at zero value that means there is no person into the region considered and grouping greater values than 0.75 $persons/m^2$ at a single bin. This could be seen as a coarse approximation, but it is enough to enclose a valid behavior to avoid those regions with high-density values.

Figure 3.9 shows the linear and angular velocities for all the persons considered as experts in the dataset, for both scenarios DS1 and DS2. The angular velocity is computed by looking at the change of orientation of the linear velocity vector between two time instants. The action space is discretized considering the behavior of experts in the dataset (see Fig. 3.9). As we are learning how to move among other people, only persons moving over certain velocity are selected as experts; the linear velocity is discretized into 8 values in $v \in [0.7\ 2.1]\ m/s$. The angular velocity is discretized in other 11 values in $\omega \in [-0.5\ 0.5]\ rad/s$. Finally, in our case, the state is used directly as features to learn the reward function.

3.6 Evaluation

By using the previous examples and the IRL algorithm, a reward function is obtained that associates a scalar value to each state. As a first evaluation of the learned reward function, we compare the actions taken by a person of the dataset with the commands given by the optimal policy obtained by solving the MDP model described above using the learned reward function.

The comparison is performed as follows: at each point of the trajectory of the selected person, the state is computed, as well as the action that should be applied according to the policy, and the actual action performed by the person. If the policy fits perfectly with the person's behavior, the actions of the MDP will be very similar to the actual ones. The actions from the MDP are not applied so that in the next point the state is the same in both cases.

In order to obtain the action given by the policy for Model 2, in which the state is divided into two substates, i.e. $\begin{pmatrix} d & \theta & \varphi \end{pmatrix}^T$ and $\begin{pmatrix} \rho_1 & \rho_2 & \rho_3 \end{pmatrix}^T$ (see Section 3.3.2), the following procedure is applied: the two substates are computed at each point so both policies are indexed by its own. This results in two vectors that contain actions likelihood, so they are multiplied among them and normalized, and finally, the most probable action is selected.

We compute the mean errors in the linear and angular velocities of each person of the dataset that was not used for training. Also, the actual actions carried out by the person are discretized to perform a fair comparison with the policy actions. Furthermore, the calculations are performed in 6 different cases based on the scenario used to obtain the expert demonstrations and the scenario used to test the policy obtained by solving the respective MDP. The first case is training with the data of scenario DS1 and evaluating in the same scenario (E1). The same case, but with scenario DS2, is denoted E2. The two next experiments evaluate the behavior obtained by training in one scenario and testing in the other one (Experiments E3 and E4). The last two experiments (E5 and E6), perform a training mixing training samples from scenarios DS1 and DS2 and evaluating the results in both scenarios respectively.

3.6.1 Policy comparison

To evaluate the results of the model presented, we first compare it with a heuristic cost based on Hall's Proxemics (PRX) theory (Hall, 1966). A cost function modeling the personal space is implemented as two Gaussian distributions as in (Kirby et al., 2009). The first function is asymmetric and placed in the front of the person with $\sigma_x = 1.20\ m$ and narrower space in the sides $\sigma_y = \sigma_x/1.5$. The second Gaussian is placed in the back of the person with $\sigma = 0.5\sigma_x$.

A new reward function is then obtained from this cost and used to determine

Table 3.1: Mean velocity errors committed by the MDP Models 1 and 2 and Proxemics-based policy regarding the expert actions. The standard errors are also showed.

	PRX closest	**Model 1 closest**	**Model 2 closest**
		Linear Vel(m/s)	
E1	0.363 ± 0.021	0.267 ± 0.018	$\mathbf{0.217 \pm 0.020}$
E2	0.371 ± 0.034	0.329 ± 0.031	$\mathbf{0.231 \pm 0.029}$
E3	0.367 ± 0.031	0.300 ± 0.029	$\mathbf{0.249 \pm 0.036}$
E4	0.279 ± 0.015	$\mathbf{0.255 \pm 0.018}$	0.269 ± 0.021
E5	0.361 ± 0.021	0.280 ± 0.019	$\mathbf{0.248 \pm 0.022}$
E6	0.269 ± 0.026	0.258 ± 0.023	$\mathbf{0.218 \pm 0.025}$
Avg	0.335 ± 0.025	0.282 ± 0.023	$\mathbf{0.239 \pm 0.026}$
		Angular Vel(rad/s)	
E1	0.094 ± 0.007	0.078 ± 0.006	$\mathbf{0.074 \pm 0.006}$
E2	0.102 ± 0.012	$\mathbf{0.100 \pm 0.012}$	0.102 ± 0.011
E3	0.094 ± 0.011	$\mathbf{0.086 \pm 0.009}$	0.096 ± 0.009
E4	0.111 ± 0.009	0.074 ± 0.007	$\mathbf{0.073 \pm 0.006}$
E5	0.095 ± 0.008	$\mathbf{0.074 \pm 0.005}$	0.091 ± 0.005
E6	0.089 ± 0.012	0.069 ± 0.008	$\mathbf{0.058 \pm 0.008}$
Avg	0.098 ± 0.010	$\mathbf{0.080 \pm 0.008}$	0.082 ± 0.008

a proxemics-based policy by solving the proposed MDP model over this reward. This way, we can compare both policies in the same framework.

The errors committed in the actions in all those approaches are presented in Table 3.1. It can be seen how the learned policies for Model 1 and Model 2 obtain, in mean, a closer behavior to the human experts than the Proxemics approach. The main difference can be observed in the linear velocity commands. In the comparison of the Model 1 and Model 2, the addition of density values to derive the policy improves the expected behavior regarding linear velocities, while angular velocities remain similar. However, there is, in general, a large deviation from the errors, so probably the models proposed cannot account for all the information used by humans to navigate among others. These results encourage us to keep tracking a better description of the social navigation task regarding feature and state spaces. We address these issues in following chapters.

3.6.2 Generalization of the model to all pedestrian

In the real world, people do not move considering just the closest pedestrian when walking through the streets. Usually, we take into account all the persons in front of us up to some meters. This is why the reward function presented previously must be completed with the information from other pedestrians in the local planning area of the robot. Although Model 2 was conceived considering the spatial distribution of people around the robot, the experimental results show that its behavior is poor concerning Model 1 (single person). This Section will evaluate another approach based on a generalization of the Model 1 to all the pedestrian in the vicinity of the robot. Thus, the previous algorithms are modified such as the action taken by the robot, in this case, is the one that maximizes the sum of the value function of the MDP for all the pedestrians on the local navigation area.

In Table 3.2, the comparison between the Model 1 considering just the closest pedestrian (closest) and its generalization to all pedestrians (all) is shown. As can be seen, there are no significant differences concerning the previous cases, and the performance is even worse in some of the cases. These results suggest that just a linear combination of features of the proposed one-pedestrian model does not account for all the necessary features to be generalized to all pedestrians case and new features should be taken into account.

In case of considering the Model 2, the generalization from one to all pedestrians is computed over the first substate $s_i = \begin{pmatrix} d & \theta & \varphi \end{pmatrix}^T$, since density values are always evaluate over all pedestrians. Taking that into account, the generalization slightly improves the results for linear velocities, whereas the angular velocity error does not improve.

Finally, Fig. 3.10 summarizes the mean errors graphically (along with the standard errors) for all the policies in the six evaluation groups. As a general conclusion, we can say that the presented models improve the results of the model with a pre-defined reward function based on Proxemics, but the differences between them are not very significant.

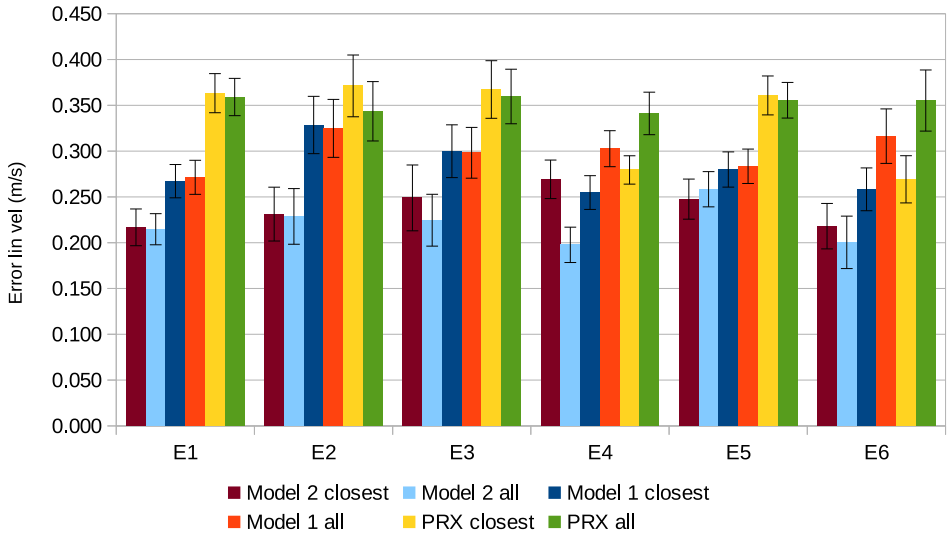
3.6.3 Transference to different scenarios

Another aspect that we evaluate is how transferable the reward function is between different scenarios (DS1 and DS2) with different conditions such as space, crowd or crossing directions of pedestrians. So, various combinations of the scenarios employed for learning and the scenarios used for testing the behavior learned are evaluated to assess the dependency on the scenario used for learning.

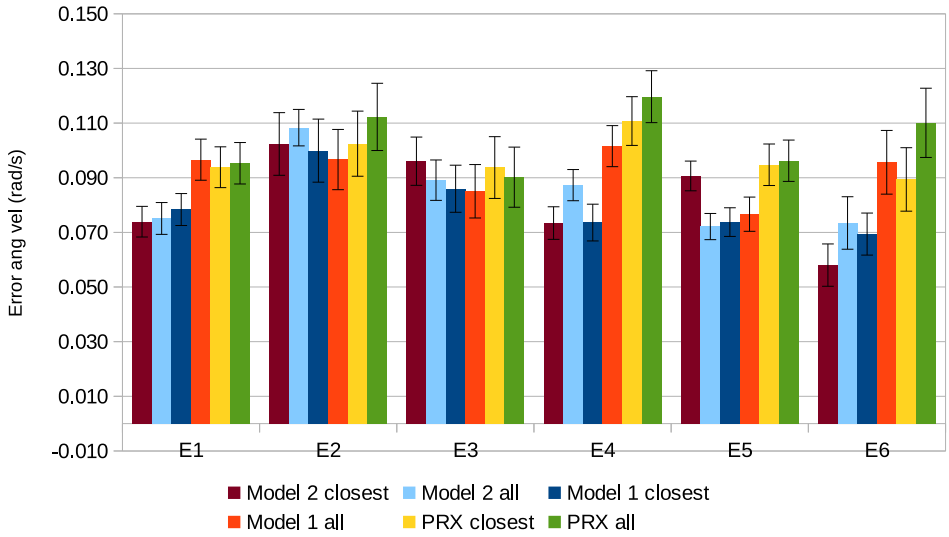
Table 3.3 shows the mean errors and standard deviations committed by the policy obtained with the best model (Model 2 closest). In this case, the scenario

Table 3.2: Comparison in mean errors of the policies of MDP Models 1 and 2 for the closest pedestrian and their generalization to all the pedestrians. The standard errors are also showed.

	Linear Vel(m/s)		Angular Vel(rad/s)	
	Model 1 closest	Model 1 all	Model 1 closest	Model 1 all
E1	$\mathbf{0.267 \pm 0.018}$	0.271 ± 0.019	$\mathbf{0.078 \pm 0.006}$	0.097 ± 0.008
E2	0.329 ± 0.031	$\mathbf{0.325 \pm 0.032}$	0.100 ± 0.012	$\mathbf{0.097 \pm 0.011}$
E3	0.300 ± 0.029	$\mathbf{0.298 \pm 0.028}$	0.086 ± 0.009	$\mathbf{0.085 \pm 0.010}$
E4	$\mathbf{0.255 \pm 0.018}$	0.303 ± 0.020	$\mathbf{0.267 \pm 0.007}$	0.271 ± 0.008
E5	$\mathbf{0.280 \pm 0.019}$	0.316 ± 0.019	$\mathbf{0.074 \pm 0.005}$	0.077 ± 0.006
E6	$\mathbf{0.258 \pm 0.023}$	0.316 ± 0.030	$\mathbf{0.069 \pm 0.008}$	0.096 ± 0.012
Avg	$\mathbf{0.282 \pm 0.023}$	0.305 ± 0.024	$\mathbf{0.080 \pm 0.008}$	0.121 ± 0.009
	Model 2 closest	Model 2 all	Model 2 closest	Model 2 all
E1	0.217 ± 0.020	$\mathbf{0.215 \pm 0.017}$	$\mathbf{0.074 \pm 0.006}$	0.075 ± 0.006
E2	0.231 ± 0.029	$\mathbf{0.229 \pm 0.030}$	$\mathbf{0.102 \pm 0.011}$	0.108 ± 0.007
E3	0.249 ± 0.036	$\mathbf{0.225 \pm 0.028}$	0.096 ± 0.009	$\mathbf{0.089 \pm 0.007}$
E4	0.269 ± 0.021	$\mathbf{0.198 \pm 0.019}$	$\mathbf{0.073 \pm 0.006}$	0.087 ± 0.006
E5	$\mathbf{0.248 \pm 0.022}$	0.258 ± 0.019	0.091 ± 0.005	$\mathbf{0.072 \pm 0.005}$
E6	0.218 ± 0.025	$\mathbf{0.200 \pm 0.029}$	$\mathbf{0.058 \pm 0.008}$	0.073 ± 0.010
Avg	0.239 ± 0.026	$\mathbf{0.221 \pm 0.024}$	$\mathbf{0.082 \pm 0.008}$	0.084 ± 0.007



(a) Errors in linear velocity.



(b) Errors in angular velocity.

Figure 3.10: Mean error bars and standard errors in the actions for all the policies considered in the groups of evaluation. 3.10a Errors in linear velocity. 3.10b Errors in angular velocity.

used for testing is the DS1, and we evaluate the policies obtained by training in different scenarios. Similar results are presented in Table 3.4, in which the

Table 3.3: Comparison of mean errors when using different scenarios for learning and scenario DS1 for evaluation.

Model 2 closest	Policy DS1	Policy DS2	Policy DS1+DS2
Linear Vel(m/s)	0.217 ± 0.020	0.269 ± 0.021	0.248 ± 0.022
Angular Vel(rad/s)	0.074 ± 0.006	0.073 ± 0.006	0.091 ± 0.005

Table 3.4: Comparison of mean errors when using different scenarios for learning and scenario DS2 for evaluation.

Model 2 closest	Policy DS1	Policy DS2	Policy DS1+DS2
Linear Vel(m/s)	0.249 ± 0.036	0.231 ± 0.029	0.218 ± 0.025
Angular Vel(rad/s)	0.096 ± 0.009	0.102 ± 0.011	0.058 ± 0.008

scenario used for testing is the DS2.

It can be observed that there are no relevant differences in the errors between scenarios, and moreover, the policy obtained from the mixed samples (DS1+DS2) does not improve the results significantly. So, the policy trained in one scenario can be correctly transferred to the other one. However, as can be observed in the data plotted in Fig. 3.5 and 3.6, the scenarios DS1 and DS2 are quite similar, so a further analysis with other different scenarios is required to evaluate the transferability of policies to other scenarios.

3.7 Conclusions

In this chapter, we analyzed the use of Inverse Reinforcement Learning based on discrete Markov Decision Processes to learn cost/reward functions from human demonstrations. We applied the method for the task of socially-acceptable robot navigation in spaces shared with humans.

Two different discrete-MDP models were proposed and the methodology to extract the cost function from a public dataset was described. Furthermore, these cost functions have been used to derive direct control policies (linear and angular velocities) for reactive motion planning of the mobile robot. These policies have been compared to the original human behavior and a policy derived from a cost function based on Proxemics theory.

The results of the derived policies comparison showed that IRL could be used to transfer some human navigation behaviors into the low-level navigation controllers of a mobile robot. The local robot controller improved the Proxemics

performance lightly and can anticipate smooth avoiding maneuvers when people are walking in opposite or almost parallel directions.

However, the general improvements were not very significant, and the social costs did not retrieve all the key-insight involved in social navigation in crowded environments. This may indicate that a richer set of features, probably including velocity features, maybe more adequate when crowds are present. However, solving a MDP becomes computationally intractable when the state space and the dimensionality are high. That led us to gross space discretization and to keep a low number of features for characterizing the task. These drawbacks found in the application of discrete MDP-based IRL techniques to the human-aware navigation problem are discussed in following chapters, and some solutions are proposed.

Chapter 4

FROG: a robust and social navigation system for crowded environments

"To Infinity and Beyond!"
Toy Story, 1995

Part of the work developed for this thesis was developed within the European FP7 project FROG (Fun Robotic Outdoors Guide)[1] (Evers et al., 2014). FROG proposed to develop a guide robot with a winning personality and behaviors that will engage tourists in a fun exploration of outdoor attractions (see Fig 4.1). The work encompassed innovation in the areas of vision-based detection, robotics design and navigation, human-robot interaction, affective computing, intelligent agent architecture and dependable autonomous outdoor robot operation.

In this chapter, the FROG project is presented, in which the work explained in Chapter 3 was integrated into the navigation system. We also emphasize the challenges in robot navigation and show the results obtained in real uncontrolled situations.

4.1 FROG project overview

The FROG project proposed the use of robotic characters in the exploration of outdoor historical and cultural sites. In particular, the Lisbon City Zoo and the Royal Alcazar of Seville were employed as a working environments for the deployment of the FROG robot. Figure 4.2 shows some pictures and characteristics of these environments.

- Lisbon City Zoo. The zoo of the city of Lisbon (Portugal) gathers around 2000 animals of 332 different species. This scenario is mostly outdoors, including non-well structured buildings and roads, slopes, grass and other challenges for robot localization and navigation.

[1] `https://www.frogrobot.eu/`

Figure 4.1: Picture of the FROG robot in the Royal Alcazar of Seville.

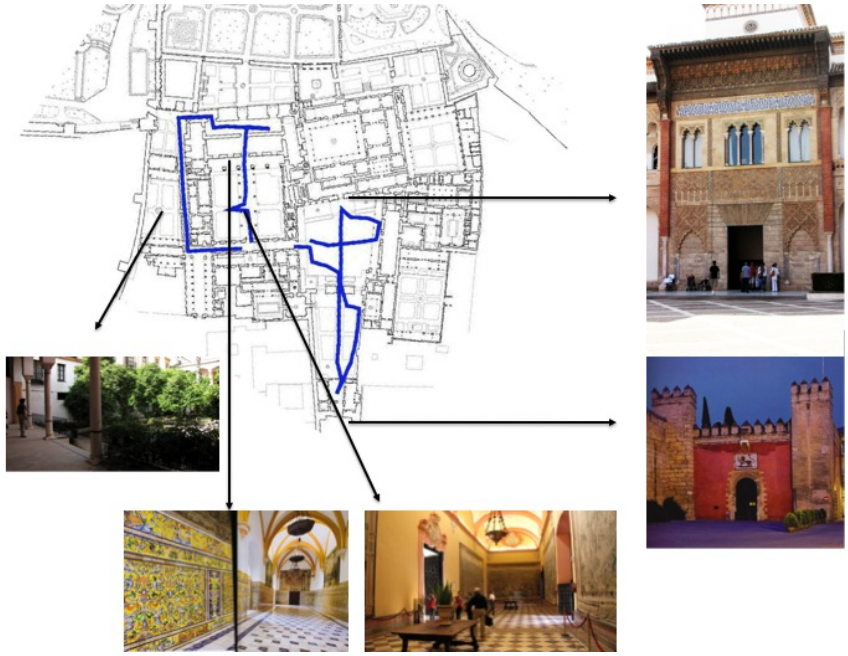
- Royal Alcazar of Seville. The Alcazar is a monumental complex located in Seville (Spain) dating from the Muslim period. It consists of a series of historical palaces from different times and styles. The Royal Alcazar has around 1.500.000 visitors per year, which means a highly crowded place combining indoors and outdoors environments.

Regarding navigation, the deployment of the FROG robot in the commented environments posed a difficult challenge. On the one hand, a very robust navigation system is required to deal with slopes, holes, changes in the terrain and several kinds of obstacles in indoor and outdoor environments. On the other hand, the system must be able to behave in a socially-acceptable manner in such crowded places with thousands of visitors.

Therefore, the objective is the development of socially acceptable, robust and efficient path planning and execution in such crowded scenarios. By efficient, we mean minimizing the amount of time when the robot is stuck due to the complexity of the scenario. By socially acceptable, human-like kind of motions when the robot is wandering or moving between way-points and objectives.

The method presented in the previous Chapter was implemented into the

(a) Lisbon city Zoo



(b) Royal Alcazar of Seville

Figure 4.2: FROG project deployment scenarios. (4.2a) The Lisbon Zoo. This scenario is mostly outdoors, including less structured buildings and roads, slopes, grass and other challenges. (4.2b) The Royal Alcazar of Seville. This scenario consists of a combination of outdoor and indoor places on man-made structures. The Royal Alcazar is a very crowded scenario.

FROG system to learn a social cost function for the robot motion. This cost function was integrated into a reactive motion planner as is explained in the following sections. The experiments and evaluations performed with the integrated system are also shown.

4.2 Navigation system

A brief description of the FROG robotic platform and the navigation tasks required in the project are described here. Moreover, the robot navigation system developed to fulfill those requirements and the integration of the social cost derived in Chapter 3 into the system is also explained in more detail.

4.2.1 Robot platform

The FROG robot consists of a skid-steering platform, with 4 wheels adapted to the scenarios considered in the project. It has an autonomy of two to four hours depending on the type of ground and the number of embedded PCs running, up to three. The robot weights 80 kg approximately and its maximum velocity is 1.6 m/s (software limited to 0.8 m/s).

The robot is equipped with a wide range of sensors for safety, localization and navigation (see Fig. 4.3):

- Odometry is computed by reading wheel encoders and angular velocities from an MTi-G IMU from XSens.
- Three laser rangefinders are considered. Two deployed horizontally forward and backward (30 m range), employed for localization and obstacle avoidance. The third laser is placed in front of the robot and tilted 45; it is used for 3D obstacle avoidance. The two horizontal lasers are also employed for people legs detection.
- A stereo camera pair is employed for person detection, pose estimation and 3D perception.
- An additional camera is used for low-range affective computing of the interacting persons.
- A sonar ring surrounding the robot.

4.2.2 Robot navigation stack

The navigation stack of the FROG robot (see Fig. 4.4) follows the classical separation between a global path planner and a local path execution module, as

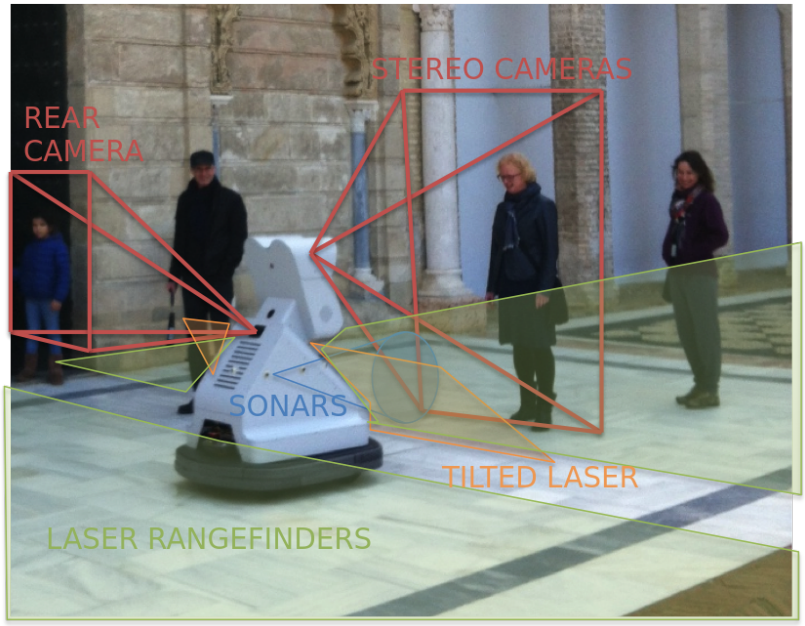


Figure 4.3: Robot platform and placement of sensors. Approximate localization and field of view of the sensors mounted in FROG robot. Green planes denote the front and rear laser scanner planes; it can be seen how the scanners cover 360 approximately around the robot. Orange plane stands for the 45 tilted laser scanner for obstacle detection. Red fields denote the field of view of the front stereo camera and back camera. Blue areas stand for sonar sensing areas.

we introduced in Section 2.2. The global planner employs the global robot pose and global models of the obstacles and potentially other models to determine a path to the goal. The local planner receives the global path and tries to follow it, by considering the most up to date sensorial information on the robot frame. This local planner generates the controls (angular and linear velocities) commanded to the robot platform.

The implementation of the navigation stack extends the navigation architecture of the Robot Operating System (ROS)[2], included in the "Fuerte" distribution. We are mainly concerned with adapting the local planner, although significant modifications were carried out to adapt the global planner to the FROG requirements.

[2] `http://wiki.ros.org/navigation`

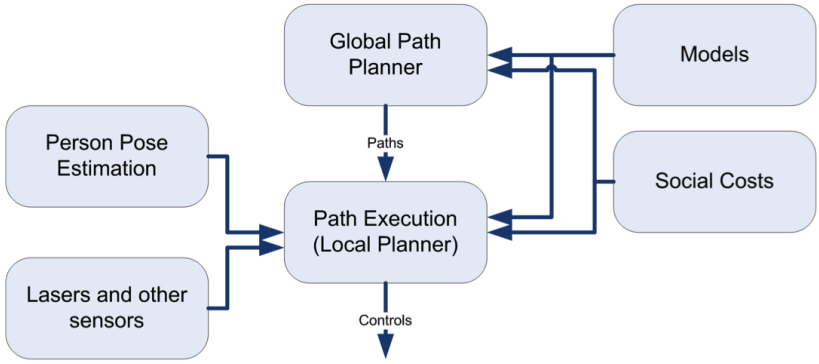


Figure 4.4: FROG navigation architecture. The navigation stack consists of a global planner, acting on global models; and a local planner acting on the most up to date information at a higher frequency.

4.2.2.1 Global path planner

The global planner is based on an A* algorithm (Hart et al., 1968) to search through the available map area and find the best path.

Many adaptations have been implemented in the global planner to fulfill FROG requirements mainly regarding efficiency. The ROS global planner[3] works well with small and medium size global maps but becomes slow when large maps (or with high resolution) are considered. Thus, persistent methods as map cleaning have been refactored and improved to work with large maps. Also, the recovery actions (actions that are taken when the robot do not find a solution for going from the current position to the specified way-point) have been refactored to perform faster and efficient actions. Also, some aspects of the global map update policy were changed.

To compute the global path and to evaluate the different trajectories to follow the path locally, two 2D grid maps are used. Each map cell of the grid map contains relevant information for motion, such as the presence of an obstacle, or membership in a recognized path. For global planning, the grid map is built from the information of the predefined navigation map along with the sensors data. This navigation map is similar to the localization map, but we include some limits in the map to restrict the areas where we do not want the robot to navigate and other information. This way, a global path is never calculated crossing these no-go areas. Further details about the mapping process and the localization system, which is not directly addressed in this thesis, can be consulted in Pérez

[3] `http://wiki.ros.org/global_planner`

et al. (2015).

4.2.2.2 "Social" local motion planner

We consider as local planner an extension of the Dynamic Window Approach (DWA) algorithm (Fox et al., 1997). The algorithm has been almost reimplemented considering computational efficiency as a major constraint. This controller predicts possible trajectories with a discrete-time simulation over a receding horizon. To ensure safe and feasible motion, the robot's kinodynamic constraints and acceleration limits have to be indicated correctly.

Another 2D grid map is built just from sensors data for local planning. This is a rectangular grid map of 8x8 meters around the robot. The portion of the global path to follow is mapped onto this area, and the grid cells are marked with distance 0 to a path point, and distance 0 to the goal. Then, a propagation algorithm efficiently marks all other cells with their Manhattan distance to the closest point marked with zero. Moreover, the sensors data is used to mark (insert obstacle information into the grid map) and clear (remove obstacle information from the grid map) merely changing the value of the corresponding cells.

The grid map information is then used by the DWA controller to choose the best trajectory among the predicted trajectories by evaluating different cost functions to balance the different robot goals. The map grid is updated with a frequency of 10 Hz to score trajectories efficiently. Therefore, the idea is to use this local motion planner, successfully employed in robot navigation, and extend it with a social cost function to endow the planner with social navigation skills. The reward cost function learned from data for the joint MDP model (model 1 + model 2), as presented in Chapter 3, is employed here as a social cost.

DWA runs by choosing a velocity (linear and angular) from a set of velocities that maximizes certain objective function in a dynamic temporal window where the forward robot movement is predicted and evaluated according to the different velocities of the set. This objective function accounts for distance to the goal (c_1), distance to the obstacles (c_2) and distance to the path that is trying to follow (c_3), as expressed in Eq. 4.1. The new social cost is also added to this objective function in this way: the features regarding the people in the robot's vicinity ($\begin{pmatrix} d & \theta & \varphi \end{pmatrix}^T$ and $\begin{pmatrix} \rho_1 & \rho_2 & \rho_3 \end{pmatrix}^T$, see Section 3.3) are computed for the projected robot positions and the corresponding reward is obtained. Then, this value is inverted and added as a new weighted term to the objective function, Eq. 4.1. Thus, the motion commands performed by the robot are modified. The weights to balance the function ($W = [\omega_1, \ldots, \omega_4]$) need to be tuned. In our approach, these values have been hand-tuned.

$$Cost = \omega_1 * c_1 + \omega_2 * c_2 + \omega_3 * c_3 + \omega_4 * \textbf{social\_cost} \tag{4.1}$$

4.2.3 Navigation tasks

The FROG robot is in charge of offering guided tours to tourists in cultural places. This requires a set of different behaviors and tasks that the robot has to be able to perform. A finite state machine (FSM) controls the robot behavior, including the different tasks involving navigation. A communication system to receive and send data between the different components of the overall FROG platform has been implemented. It is based on JSON lightweight data-interchange format.

In particular, the navigation and localization system, besides continuously publishing the location of the robot to the other modules, needs to receive the navigation commands from the top-level FROG behavior tree (FSM). Moreover, it has to communicate the execution state of the navigation action commanded.

The interface defined by the navigation stack involves the following actions:

- Going to a defined goal. The location and the desired orientation of the goal point are received. Then, the path to the goal is calculated and the navigation is initiated.

- Rotating to reach a specific orientation. This behavior is similar to the previous one, but in this case, the location of the goal is the current location of the robot causing only a turning movement.

- Approaching to a particular person. The identification number of the targeted person is used instead of the goal coordinates. Moreover, a value of safety distance is also indicated to stop the robot in front of the person keeping the stated range between them. The approaching maneuver is done taking into account the orientation of the person, so the robot tries to face the person. Furthermore, we perform a simple tracking of the person by using a time window. This way, we can change the navigation goal if the person is moving.

- Rotating to face a particular person. We only consider the orientation of the targeted person to rotate the robot to face the person.

- Cancel the current navigation execution and stop the robot. In this case, the navigation to the current goal is aborted, and the robot movement is stopped.

- Docking and undocking. It commands the robot to look for visual markers placed on the charging station and perform the docking (or undocking) maneuver.

Also, a specific navigation behavior has been programmed to approach the desired goal with the best accuracy permitted. This behavior consists of decreasing the velocities approaching the goal point. Similar behavior has been included to initiate the robot movement smoothly, starting with low velocity and increasing it to reach the commanded velocity.

4.3 Navigation experiments

This section shows all the experiments and evaluations of the integrated navigation system in real controlled scenarios and the final environments such as the Zoo of Lisbon and the Royal Alcazar of Seville.

4.3.1 Quantitative experiments

A differential-drive Pioneer robot platform is employed to perform a set of controlled experiments in real environments with human confederates. Figure 4.5 shows an image of the outdoor scenario at the Pablo de Olavide University where the experiments were performed. In these tests, the robot autonomously navigates from a starting point to a given waypoint, encountering persons in its path during the execution.

We compare the traditional ROS navigation system without any social component (we call it "No social") with the following approaches that add an extra-cost to the DWA objective function:

- Cost based on Proxemics (PRX closest).
- Cost based on the cost function learned with the MDP model 1 (Model 1)
- Cost based on the cost function learned with the MDP model 1 and extended to all the pedestrians in the robot's vicinity (Model 1 all).
- Cost based on the cost function learned with the MDP Model 2 of people densities in regions in front of the robot (Model 2).
- Cost based on the combination of the cost function learned with the MDP models 1 and 2 (Model 1 - 2).

Finally, the following metrics are employed for comparison: the total distance traveled towards the goal (TD), the time employed to reach the waypoint (T) and the minimum and medium distance to the persons (Min PD and Mean PD respectively) along the path. By computing the execution time and total distance traveled we can assess the effectiveness to reach the goal, while the distances to persons let us know how the personal space is conserved and its effect in the deviation distance from the global path.

Figure 4.5: Capture of the outdoor scenario for experiments at the Pablo de Olavide University.

4.3.1.1 Experiments with static pedestrians

In this experiment setup, the robot has to cross a controlled scenario with pedestrian standing, talking to each other. It is assumed a static scenario in the sense that people do not move from their initial position (see Fig. 4.5). We have performed four runs for each approach, keeping the same configuration among the different runs.

The spatial disposition of the three static pedestrians employed in the experiment is presented in Fig. 4.6. An example of the trajectories performed by the robot by using different approaches is also showed. At first sight, it can be seen some differences between the trajectories depending on the approach employed. However, we will better explain the results by analyzing all the tests performed in the experiments, which are presented in the table 4.1:

- No social. This navigation system without the social component tries to optimize time and distance traveled. So, the average distance and minimum distance to the pedestrians is the lowest of all the approaches.

- PRX closest. This classical approach improves a bit the results of the "No social" navigation. However, it performs a too significant avoidance when the robot is close to pedestrian which is considered unnecessary. The

Table 4.1: Experimental results in a static scenario.

	T (s)	TD (m)	Mean PD (m)	Min PD (m)
No social	$\mathbf{50.65 \pm 0.21}$	21.76 ± 0.29	2.82 ± 0.06	0.82 ± 0.03
PRX closest	71.00 ± 1.55	21.09 ± 1.04	3.69 ± 1.69	1.05 ± 0.18
Model 1	54.28 ± 3.39	$\mathbf{20.50 \pm 0.22}$	5.19 ± 1.06	1.29 ± 0.18
Model 1 All	58.85 ± 0.21	22.68 ± 0.19	$\mathbf{5.22 \pm 1.69}$	$\mathbf{1.69 \pm 0.04}$
Model 2	51.05 ± 0.92	24.53 ± 0.01	4.50 ± 0.21	0.87 ± 0.02
Model 1 - 2	53.27 ± 3.17	21.19 ± 1.56	4.00 ± 1.82	1.01 ± 0.29

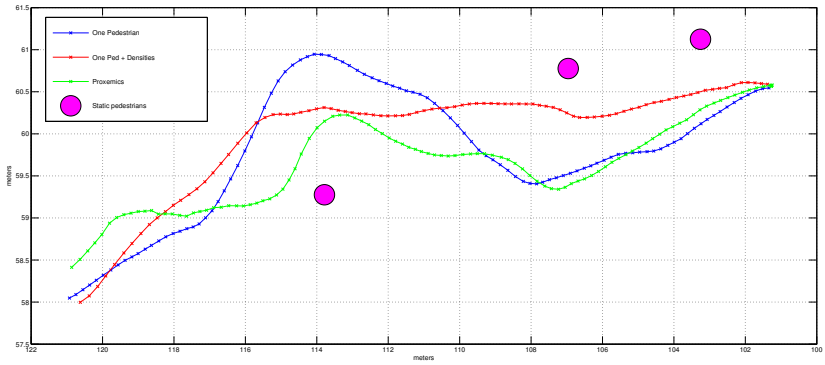


Figure 4.6: Example of the trajectories performed by the robot according to the different DWA objective functions.

excessive execution time may be due to inefficient programming of the evaluation of the possible trajectories, which is a critical point regarding execution time.

- Model 1. The application of the reward function derived from this MDP model improves the results of the Proxemics approach. It can smoothly modify its trajectory to cross a walking person in the opposite direction. The behavior of this model can be suitable for no-crowded scenarios, but the performance can decrease if several people are surrounding the robot.

- Model 1 All. According to the results, the generalization from one pedestrian to all has a similar performance to the model for one pedestrian. These results can be different in experiments with more people. Nevertheless, we think that this generalization cannot retrieve the necessary key

Table 4.2: Experimental results in a dynamic scenario

	T (s)	TD (m)	Mean PD (m)	Min PD (m)
No social	$\mathbf{59.22 \pm 0.06}$	20.38 ± 0.29	2.31 ± 0.15	0.20 ± 0.09
PRX closest	68.44 ± 0.02	20.99 ± 0.68	2.59 ± 0.10	0.38 ± 0.04
Model 1	69.45 ± 7.01	21.85 ± 1.89	$\mathbf{4.46 \pm 0.19}$	0.20 ± 0.17
Model 1 All	61.20 ± 0.0	20.50 ± 0.0	2.49 ± 0.16	$\mathbf{0.60 \pm 0.12}$
Model 2	65.26 ± 10.08	20.75 ± 0.24	2.56 ± 0.17	0.47 ± 0.16
Model 1 - 2	60.20 ± 1.81	$\mathbf{20.36 \pm 0.35}$	2.46 ± 0.10	0.33 ± 0.02

aspects for good navigation in crowded environments.

- Model 2. As we can observe in the table, in this case, the distances to pedestrians are lower than the case of model 1 and its generalization. However, it is not taken into account aspects of the orientation or movement direction of the pedestrian what can bring unwanted avoiding behaviors.

- Model 1 - 2. This approach can anticipate the crossing maneuvers better than Proxemics, and therefore, without performing excessive avoiding movements. Moreover, it keeps a shorter distance to the pedestrians than model 1 approaches without being uncomfortable for them. We consider that this behavior can be the more appropriate for crowded environments.

4.3.1.2 Experiments with dynamic pedestrians

In the dynamic scenario, we use the same scenario than before, but in this case, the pedestrians are moving and crossing the area. Moreover, we will show results over another non-controlled scenario as the main corridor at Pablo de Olavide University.

First, we show the results in a controlled scenario, where we can repeat the experiment in equal (or nearly equal) and controlled conditions. We performed 4 runs on each navigation approach. Secondly, we run the joint model (model 1 and model 2) approach in an uncontrolled scenario. This place is the main corridor of the Pablo de Olavide University, where a lot of students cross every day.

The set up of the controlled experiment is as follows. Two pedestrians walk opposite to robot direction forcing it to avoid them. Then, other pedestrian crosses in diagonal in front of the robot. Finally, two new pedestrians pass the robot on the left walking in the same direction as it.

Figure 4.7: Left: FROG guiding the reviewers and project members at Lisbon Zoo. Right: an overall view of the performed tour on the map.

The metrics values are presented in Table 4.2. As can be seen, the results are very similar to those obtained in case of static pedestrians (Table 4.1). Again, all the "social" approaches suggested improved the "no social" behavior. In this case, the model 1 seems to make some excessive avoiding maneuvers. Although the results are not very conclusive and some approaches obtain similar metrics values, we think the mixed approach (Model 1- 2) shows a balanced behavior. It can plan a good path regarding time efficiency and distance traveled (T, TD) as well as keeping acceptable distances to pedestrians (Mean PD, Min PD).

Regarding the experiment in the corridor of the University. It lasted about 12 minutes, and the traveled distance was 221.18 m. The average distance to the pedestrians was 4.2 m. The behavior of the navigation approach was satisfactory according to the results obtained in the controlled experiments. The actions taken to avoid the crossing pedestrians were smooth and sociable well-accepted by the pedestrian.

4.3.2 Qualitative experiments in final scenarios

The data collected during the different experiments and demonstrations performed in the working environments of the project are presented here.

4.3.2.1 Experiments in Lisbon Zoo

In the second-year demo (2013), the FROG robot guided visitors around in the Lisbon City Zoo, showing the animals, and telling about the species, their normal behavior and the natural environment (see Fig. 4.7). The demo involved a **route of 750 meters**, with different points of interest, and it lasted **45 minutes** approximately.

The whole demo week the robot was navigating autonomously for more than 4 *kms*. This demo, where the navigation functionalities were not using the social component, demonstrated its robust and accurate localization for navigation in such challenge outdoor scenario.

4.3.2.2 Experiments at the Royal Alcazar

During the third and last year of the FROG project, additional navigation experiments were performed at the final demo site, the Royal Alcazar in Seville, during June and September of 2014.

Figure 4.8 shows the typical mission performed. It involves an undocking maneuver, a tour including 7 Points of Interest, the way back to the charging station and docking again onto the charging station. In these experiments, all the social navigation functionalities were activated.

In the experiments, the robot was performing the complete tour, navigating to all the Points of Interest and showing the proper content. In June, the guided tour had an average distance of 460 meters approximately, including the return path to the initial point, which involves a slightly different route. During 8 days of continuous testing, the tour was performed twice a day on average, which gives a total of nearly 7.4 *kms* of autonomous navigation in a very crowded scenario. A summary of the data collected along these tours is presented in Table 4.3. The total duration of the tour in minutes, the total distance traveled by the robot (meters), and the average distance between the robot and the pedestrians (meters), are shown.

Table 4.4 shows the collected data of some missions completed or nearly completed that were performed in the experiments in September 2014 at the Royal Alcazar of Seville. The tour was slightly modified concerning the tours performed in June. The location of some points of interest was changed and new content was added. The tour had an average distance of 500 meters, although some slightly different routes were also tested. This explains the differences in total time and distance traveled between some tests, besides the different crowd densities encountered. Furthermore, the person tracking with lasers was improved and integrated, so a new data column expressing the total number of pedestrians detected by the laser-based tracker is included in the Table. These values are higher than the real number of pedestrians due to some false positives and person re-entering in the detection area.

Finally, a summary of the total number of missions performed in the experiments performed in June and September, along with the percentage of successfully-completed tours is shown in Table 4.5. Regarding the robustness of the navigation stack (81% of success), only in 1 case of the 26 missions, manual intervention was required to recover the robot from a blocking situation. The main patio was

Figure 4.8: An overall view of the performed tours in the Royal Alcazar of Seville. Three different examples of the trajectories for the tour are shown in red, green and blue (the tour details depend on the time of the day, leading to some differences in the trajectories). Different photographies illustrate the different zones in the trajectory.

occupied with tables and tablecloths for a special gala dinner at the Royal Alcazar. Normally this patio is free, and there are no permanent obstacles. The robot was able to negotiate the (previously unmapped) obstacles of the place, even though the remaining space was quite narrow, reaching the corresponding Point of Interest in the square. When leaving the tables zone, the wind provoked the tablecloths began flapping, coming into the robot safety buffer zone, and even touching the robot. The robot thus got stuck, and a manual control was required to overcome the situation. The rest of failures were caused by hardware problems (mainly related to the inertial sensor (IMU)).

Table 4.3: Data from several of the tours performed at the Royal Alcazar with the FROG robot in June 2014.

Complete tour			
Date and Time	T (min)	TD (m)	Mean PD (m)
19/06/14 15:02	24.70	223.30	3.42
20/06/14 17:52	21.92	258.23	2.65
26/06/14 11:53	26.60	257.91	2.09
26/06/14 17:35	27.73	275.46	2.25
Complete tour and trip back to charging station			
21/06/14 12:40	38.03	460.22	2.23
22/06/14 16:52	37.39	453.40	2.16
23/06/14 11:55	37.08	468.25	2.30
23/06/14 16:46	35.93	453.77	2.24
25/06/14 18:08	36.51	458.40	2.36
26/06/14 11:08	36.02	434.97	2.20

4.4 End-users evaluations

Besides the quantitative results presented previously, here we show a qualitative sociological evaluation. In particular, to gain an understanding into how people experience a robot guided tour, some visitors to the Royal Alcazar who followed (parts of) a tour given by the FROG robot were observed and interviewed. These studies were conducted by the Human-Media Interaction group, from the University of Twente, as a partner of the FROG project (Karreman et al., 2014; Karreman, 2016).

The evaluations of user experience with the FROG robot were performed in the last week of June 2014. The FROG robot gave 16 autonomous tours in seven days, many of which were previously presented in Table 4.3. Four of the 16 tours were with recruited visitors, and the other 12 were given to naive visitors of the Royal Alcazar. The study focused on the four tours with recruited visitors. People participating in these tours evaluated the user experience of the robot throughout the tour. However, we also evaluated the tours of non-recruited visitors. For these tours, observations were made during the tours and interviews after the tour had ended.

We show here only the comments and information received from the visitors

Table 4.4: Data from several of the tours performed at the Royal Alcazar with the FROG robot in September 2014.

Date and Time	T (min)	TD (m)	Mean PD (m)	Pedestrians
18/09/14 10:22	40.09	440.22	2.89	1881
22/09/14 11:20	53.49	558.74	2.52	3599
22/09/14 18:41	59.22	685.33	2.51	2942
23/09/14 10:25	53.57	540.96	2.47	3108
23/09/14 15:18	46.59	492.84	2.37	2246
23/09/14 17:33	39.59	461.94	2.37	2202
24/09/14 09:49	57.02	456.52	3.01	3283
24/09/14 16:57	43.96	510.28	2.53	2799
25/09/14 10:27	40.25	390.46	2.46	2628
25/09/14 16:41	49.63	592.93	2.62	3104
26/09/14 11:00	35.98	354.47	2.49	1464
26/09/14 18:47	56.16	520.76	2.05	3918
Average	47.96 ± 7.94	500.45 ± 89.84	2.52 ± 0.24	2764.50

Table 4.5: Percentages of successful missions performed in the experiments in June and September 2014 at the Royal Alcazar

Number of missions	% successful tours
26	81

related to the navigation experience. Also, one or two quotes are presented in each paragraph to illustrate the visitors' experiences in their own words.

"And we were flabbergasted, we thought that the man with the controller was controlling the robot..."

At the beginning of the tour, participants walked carefully behind the robot, because they needed to get used to the robots movements. In the interviews they stated that the behavior of the robot was clear to them. However, most recruited visitors mentioned during the whole tour that they were unsure where the robot was going or if they were standing in its way when it started driving. Participants stated that the robot drove a bit jerkily and stopped now and then, so they had to watch out not to be hit by the robot or bump into the robot. Due to this, some participants chose to walk next to the robot.

"You need to get used to following the robot, it is jerky and so of course, so you need to get used to it, because where does it go?"

"Ok, to which side..."

Participants mentioned that the robot was a bit slow while driving. Some participants mentioned that for larger groups the speed might be alright because larger groups tend to move a bit slower. When the robot was close to the Point of Interest, the robot needed a lot of time to find a place to settle. Also, at the Point of Interest, participants would appreciate if the robot could turn towards them before turning to the final position to start the story because some participants felt ignored by the robot.

When the robot was moving, it gave too little feedback to the followers. A map and some text were displayed on the screen, but for those people who followed the robot and walked next to or behind the robot that information was not visible. Therefore, it was sometimes unclear to the participants where the robot was going. To obtain this information, visitors sometimes walked to the front of the robot to look at the screen. One recruited participant thought that having the robot driving backward would improve this point.

"I felt best walking beside it"

As a general conclusion, the visitors were happily impressed about being guided by an autonomous robot. They felt safe walking with the robot, and they did not see it as a threat. The most of negative comments posed the difficulties of the robot to correctly show its intentions to the visitors when it tried to initiate the navigation to the next point and the intention to stop at

the points of interest to start a new explanation. It is worthy to mention that these issues are not considered in the navigation function learned, which is only referred to the navigation itself, and thereby, the navigation behavior seems to be socially accepted by the visitors. A new social function could be considered to improve the negative behaviors commented.

These drawbacks are also related to the robot kinematics. We detected that the differential motion in such a big and heavy robot turned out in slow and stiff rotations, mainly in rotations in place, in most of the ground types. This caused a negative effect on the robot movement since the robot was not able to react quickly and accordingly to the movements of the people crowd around it.

4.5 Conclusions

This Chapter summarizes the robot navigation system developed in the framework of the European FROG project. An efficient and safe navigation system has been implemented according to the FROG system specifications, paying special attention to the socially-acceptable behavior of this navigation.

The approach presented in Chapter 3 is integrated into the reactive motion planner of the navigation system of the FROG robot. It makes use of Inverse Reinforcement Learning to learn cost/reward functions from examples for the task of navigating among persons.

The chapter also summarizes the navigation performed by the robot during the second year review at Lisbon Zoo. The robot was able to navigate within a very complex scenario, with many people surrounding the system without problems. Although the social skills presented in this document were not evaluated at the Lisbon Zoo, the rest of the navigation system (localization, collision avoidance, smooth navigation, Point of Interest integration, etc.) were successfully tested in more than 4 Kilometers of fully autonomous robot navigation. Moreover, results obtained at the Royal Alcazar planned for June, and September 2014 are included. This time, all the social navigation skills were employed. During the sessions, more than 7.4 km in autonomous mode, guiding persons in a very challenging and crowded environment was successfully performed.

However, throughout that extensive experimentation with the real robot in real uncontrolled and crowded scenarios, we detected some drawbacks and extracted some conclusions:

- First, the robot kinematics was a drawback. A big and heavy platform with a 4-wheels differential motion was not able to give a quick-enough response in those dynamic and highly populated environments. Therefore, a lighter robot and/or a holonomic platform that allows quicker reactions and faster movements would be more convenient.

- Second, the designed cost functions were not able to capture all necessary insight to perform socially-acceptable navigation. That led the robot to behave correctly just in some particular situations. The limitations of the discrete MDPs commented in Chapter 3, did not allow to learn a more complex cost function for the general task. This issue motivated us to employ a different planning method, like the sampling-based planners combined with different LfD techniques, with the aim of overcoming some of these drawbacks. This will be addressed in the following chapters.

Chapter 5

Imitating local human-robot interactions based on sampling-based planners

"I'm having an old friend for dinner."
The Silence of the Lambs, 1991

This chapter continues the study of the LfD techniques applied to the human-aware navigation problem. With the aim of overcoming some of the drawbacks experimented in the use of MDPs, as explained in Chapter 3, we decided to use a more flexible motion planner, like the sampling-based algorithms. Thus, we combine this planner with LfD techniques to endow it with social normative behavior.

In particular, we present a framework for modeling human-robot interactions based on unsupervised learning and sampling-based planners for execution. The approach makes use of Gaussian Mixture Models (GMMs) to model the physical interaction of the robot and the person when the robot is teleoperated or guided by an expert. The learned models are integrated into a sampling-based planner, a RRT*, at two levels: as a cost function to plan trajectories considering behavior constraints, and as configuration space sampling bias to discard samples with low cost according to the behaviors. The algorithm is successfully tested in the laboratory using an actual robot and real trajectories examples provided by an expert.

5.1 Introduction

Many novel approaches for human-aware navigation are based on Learning from Demonstrations methods (LfD) (Argall et al., 2009), to learn socially acceptable behaviors from real data collected under various social situations, avoiding the need of a handcrafted explicit formulation of the behaviors.

As part of these methods, Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000), as introduced in Chapter 3, is an excellent candidate to derive

such models. We made use of IRL in that Chapter and publications (Ramón-Vigo et al., 2014; Pérez-Higueras et al., 2014) to learn local social navigation policies. Also, it is employed in (Islas-Ramirez et al., 2016) to learn how to approach people for interaction; a task that will be addressed here too.

However, IRL approaches typically make use of discrete Markov Decision Processes (MDPs) as the underlying process. General problems are complex to be encoded with MDPs due to its computational complexity (Okal et al., 2015). Unlike the work presented in Chapter 3, we present here a first approach that makes use of state-of-the-art sampling-based planners, as optimal Rapidly-exploring Random Trees (RRT*), (Karaman and Frazzoli, 2011), to be able to work on continuous configuration spaces with higher dimensionality. These planners already reason about obstacles present in the environment, and the goal is to incorporate into them information about the social task at hand from data.

Furthermore, we employ here a different LfD approach for human-aware navigation based on Gaussian Mixture Models (GMMs). They offer a flexible framework to model the relationships between the relevant features that arise when the robot is performing a particular navigation task. GMMs are a well-suited representation for unsupervised extraction of continuous feature distributions, and it has also shown its utility as a model for robot skills representations in Programming by Demonstration (PbD) settings (Calinon, 2009).

In (Claassens, 2010), the authors present a PbD framework in which GMMs are used to retrieve the statistical constraints of several demonstrations of a particular task, like the approach in (Calinon and Billard, 2008). After that, these constraints are used to guide a sampled-based planner based on RRT. Different from these approaches, we aim to go a step further and use GMM to incorporate social navigation behavior into a cost-based RRT* planner. The goal is to find a safe path which imitates a behavior by remaining within statistically determined constraints. For doing this, we propose, first, to bias the RRT* random samples towards the regions of the configuration space that comply with the model of the task extracted from data. And second, including a new cost function into the RRT* planner to better account for paths that follow the learned behaviors. At the same time, this permits better generalization to new situations by still finding short length paths for different conditions.

In the TERESA Project[1] (Shiarlis et al., 2015), telepresence robots are enhanced to navigate autonomously inside buildings. The project considers the development of techniques for safe and efficient obstacle avoidance while reaching navigation goals. This task becomes challenging when people are considered and explicitly modeled into the navigation approach (see Chapter 7). We can leverage the fact that we have the telepresence robot of the TERESA Project

[1] `http://teresaproject.eu/`

Figure 5.1: Example pictures of the TERESA telepresence system.

so that we can extract useful information from the users controlling the robot. An example of this setup is shown in Fig. 5.1 where the controller of the telepresence system and the robot interacting with other person are shown. Using examples of (teleoperated) robot paths and the relevant configurations (features) of the performed task open the door for extracting the relevant relations or constraints that best represent such kind of paths. The main hypothesis here, and the whole thesis indeed, is that these paths enclose the social implications that a human takes into account when he is performing such task, at least at the same situations performed in the experiments.

In summary, this chapter proposes an approach to obtain, in an unsupervised way, these human-robot interaction models from the data gathered by the TERESA telepresence robot. Then, we employ these models combined with a sampling-based planner to reproduce the learned behavior efficiently. While the method is general, we present results in some particular tasks involved in social interaction in navigation.

5.2 Interaction tasks proposed

The proposed approach of imitation learning can be employed to model specific local interactions between the robot and the people for certain tasks. These models will be then used by the local RRT* planner that will provide more flexibility to deal with possible obstacles in real situations since obstacles are not considered in the models learned. To illustrate the proposed approach we focus on two particular tasks:

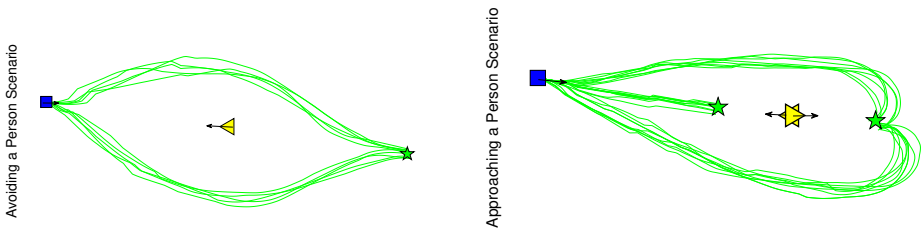


Figure 5.2: Example tasks to be encoded using GMMs. The initial robot position is indicated with a blue square and the goal with a green star. The person position is indicated by a yellow triangle, and the trajectories are shown in green lines. Avoiding a standing person is shown in the left image. Approaching a person in two cases, facing the robot or back to the robot, is depicted on the right image.

- **Avoiding**. The robot avoids a standing person that is facing it. The avoidance maneuver can be performed by passing through the left or the right side.
- **Approaching**. The robot approaches a standing person in an arbitrary orientation. For the sake of simplicity, we study two particular cases: when the person is looking to the robot, and when the standing person is back to the robot. In this latter, the robot can approach the person by the left or right side. This "social" approaching task is addressed in the TERESA project, in which the generalization of the method to cover all the rest of possible approaching angles and the integration and use in the TERESA telepresence system is explained in Chapter 7.

Figure 5.2 shows a graphical illustration of some example trajectories employed for model learning of the two tasks proposed.

5.3 GMMs for interaction modelling

A proper choice of the relevant features $\mathbf{f} = [f_1, f_2, \ldots, f_n]^T$ when encoding a particular navigation task is crucial, as it provides part of the solution to the problem of defining what is important to imitate. For the work developed here, we have considered as features the distance (d) and the relative angle (θ) between the robot and the person in the scene, logged with timestamps. The angle values are continuous measures from $[-\pi, \pi)$, negative clockwise (concerning the person's gaze). Thus, a set composed by N datapoints $\zeta = \{\zeta_j\}_{j=1}^{N}$ of $D = 2$

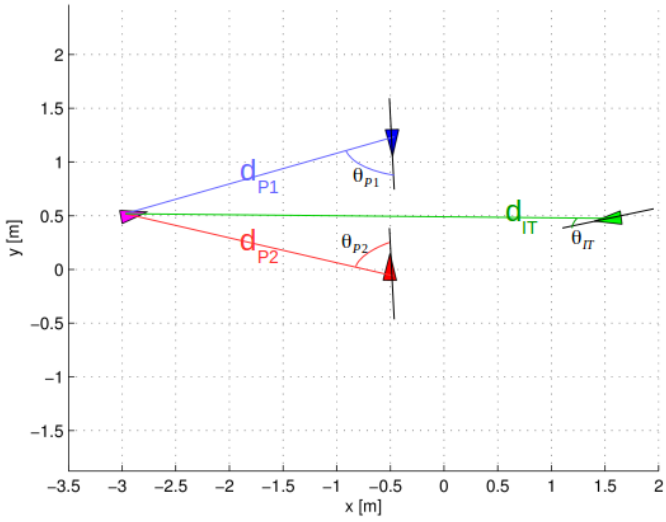


Figure 5.3: Illustration of the features chosen to encode the proposed tasks through GMMs. Representation of the features d and θ of three people represented by the blue, red and green triangles according to the robot position represented by a pink triangle.

dimensions is considered. Figure 5.3 shows an illustrative example of the features considered (d, θ), in which the features of three people (blue, red and green triangles) regarding the robot (pink triangle) are depicted.

The time feature is left out because the dynamics of the tasks proposed is not taken into account. That means that only static scenarios, in which the pedestrians stand still in their initial positions, are considered as demonstrations in this work. However, this is not a limitation of the method, and time could be considered as feature describing these tasks or other tasks.

According to our work in (Ramón-Vigo et al., 2015) and the social psychology studies of the movement and interaction of the pedestrians in (Dennis Middlemist et al., 1976; Bitgood and Dukes, 2006), the features (d, θ) considered here allows us to model the tasks at hand since we are considering learning from static scenarios. However the selection of features is not a limitation of this technique, and other features can be added for more complicated tasks, including those that can describe the dynamics of the task execution, like time and velocities.

From the demonstrated trajectories of the features $\mathcal{D}$, it is possible to extract a GMM, so that the probability of a particular combination of feature values is given by:

$$p(\mathbf{f}|\mathcal{D}) = \sum_{i=1}^{k} \omega_i \mathcal{N}(\mathbf{f}; \mu^i_{\mathcal{D}}, \Sigma^i_{\mathcal{D}}) \tag{5.1}$$

with k Gaussian *modes*. The GMM is then described by the set of parameters $\left\{\omega_i, \mu^i_{\mathcal{D}}, \Sigma^i_{\mathcal{D}}\right\}_{i=1}^{K}$, respectively representing the *prior* probabilities, *centers* and *covariance* matrices of the model. The *prior* probabilities, ω_i, satisfy $\omega_i \in [0, 1]$ and $\sum_{i=1}^{K} \omega_i = 1$. Each *mode* of the GMM is characterized by a multi-variate Gaussian distribution with mean μ^i and covariance matrix Σ^i:

$$\mu^i = \begin{bmatrix} \overline{f}^i_1 \\ \overline{f}^i_2 \\ \vdots \\ \overline{f}^i_n \end{bmatrix} \tag{5.2}$$

$$\Sigma^i = \begin{bmatrix} \Sigma^i_{11} & \Sigma^i_{12} & \cdots & \Sigma^i_{1n} \\ \Sigma^i_{21} & \Sigma^i_{22} & \cdots & \Sigma^i_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma^i_{n1} & \Sigma^i_{n2} & \cdots & \Sigma^i_{nn} \end{bmatrix} \tag{5.3}$$

The GMM parameters are learned from the demonstration data through the use of the *Expectation-Maximization* (EM) algorithm (Dempster et al., 1977) that is seeded with an initial estimate of density centers calculated with the k-means algorithm. A drawback of EM is that the optimal number of components k in a model may not be known forehand. One usual criterion for model selection is the Bayesian Information Criterion (BIC) (Schwarz, 1978), which performs a trade-off between how well the model fits the data and a penalty factor aiming at minimizing the number of parameters. The main drawback of BIC is that it requires the estimation of multiple models. So, the best number of components that can fit the demonstrated example has been tested empirically in the experiments performed here.

5.4 The reproduction planner

To reproduce the behavior demonstrated in the tasks, we provide a sampling-based motion planner (a RRT*) with the information encoded in the learned GMMs. This RRT* planner is extensively used in this thesis, so we present here a detailed description of the algorithm first. Then, we explain the modifications performed in the algorithm to use the learned models.

5.4.1 RRT* planner

Optimal Rapidly Exploring Random Trees, known as RRT*, (Karaman and Frazzoli, 2011) is a technique for (asymptotical) optimal motion planning. It considers that a cost function is associated with each point x in the configuration space $\mathcal{X}$. The RRT* seeks to obtain the trajectory ζ^* that minimizes total cost along the path $C(\zeta)$. It does so by randomly sampling the continuous configuration space and creating a tree towards the goal, indicated as $G = (V, E)$, with V the vertices of the tree and E the edges that connect the vertices.

The general procedure to build the tree is the following: for each new sample, an edge is created from the tree vertex (belonging to the group of the "closest" tree vertices to the sample) that can be connected to the new sample along a path with minimum cost. So, the new sample becomes a new vertex of the tree. Then, a rewiring process of the tree is performed, in which new edges are created from the new vertex to the vertices in the previous nearest-vertices group. If the path through the new vertex has a lower cost than the path through the current parent; in this case, the edge linking the vertex to its current parent is deleted, to maintain the tree structure. The process does not finish when a path from the start configuration to the goal configuration is found, but when a maximum number of samples has been drawn and/or a time limit has expired.

A detailed description can be seen in Algorithm 5.1. Next, we explain the most relevant procedures and functions involved:

- $\mathbf{SampleFree_i}(\mathbf{x_{goal}}, \epsilon)$. This function returns a valid sample (obstacle free) of the configuration space. The samples are drawn from a uniform distribution. Moreover, with a low probability, indicated by ϵ, the goal configuration x_{goal} (or region X_{goal}) is sampled.

- $\mathbf{Nearest}(\mathbf{G}, \mathbf{x})$. Given the current tree G and a sample x of the configuration space $\mathcal{X}$, this function returns the configuration point $x_{nearest}$ of the tree G that is the "closest" to the sample x, according to some given distance criteria, such as Euclidean distance or Manhattan distance among others.

- $\mathbf{Steer}(\mathbf{x_n}, \mathbf{x_m})$. Given two points $x_n, x_m \in \mathcal{X}$, this function returns a point $z \in \mathcal{X}$ such that z is "closer" to x_m than x_n is. The point z minimizes $\|z - x_m\|$ while at the same time maintains $\|z - x_n\| \leq \eta$, for a pre-specified $\eta > 0$. In other words, the *Steer* function moves the robot (in straight line or with kinodynamic constraints) from the point x_n in the direction of x_m until reaching the point or a maximum distance indicated by η.

- $\mathbf{CollFree}(\mathbf{x_n}, \mathbf{x_m})$. This binary function returns whether the path between the points x_n and x_m is free of collision with any obstacle or not. It can

Algorithm 5.1: RRT*

Input: Start configuration x_{init}; goal configuration x_{goal}
Output: Tree $G = (V, E)$;

$V \leftarrow x_{init}$; $E \leftarrow \emptyset$;
for $i = 1$ to $maxSamples$ **do**
  $x_{rand} \leftarrow SampleFree_i(x_{goal}, \epsilon)$;
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand})$;
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Near(G = (V, E), x_{new}, radius)$;
    $V \leftarrow V \cup \{x_{new}\}$;
    $x_{min} \leftarrow x_{nearest}$;
    $c_{min} \leftarrow Cost(X_{nearest}) + MotCost(x_{nearest}, x_{new})$;
    for each $x_{near} \in X_{near}$ **do** // Connect along a minimum-cost path
      if $CollFree(x_{near}, x_{new}) \wedge Cost(x_{near}) + MotCost(x_{near}, x_{new}) < c_{min}$ **then**
        $x_{min} \leftarrow x_{near}$;
        $c_{min} \leftarrow Cost(x_{near}) + MotCost(x_{near}, x_{new})$;
      end if
    end for
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
    for each $x_{near} \in X_{near}$ **do** // Rewire the tree
      if $CollFree(x_{new}, x_{near}) \wedge Cost(x_{new}) + MotCost(x_{new}, x_{near}) < Cost(x_{near})$ **then**
        $x_{parent} \leftarrow Parent(x_{near})$;
      end if
      $E \leftarrow (E \setminus \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$;
    end for
  end if
end for

perform the checking by considering the path as a straight line connecting the points or the resulting path taking into account the kinodynamic constraints of the platform (as a combination with the *Steer* function).

- **Near**$(\mathbf{G}, \mathbf{x}, \mathbf{r})$. Given the point $x \in \mathcal{X}$, the function $Near$ returns the group of points (vertices) X_{near} of the tree G that are within a ball of radius r around the point x. The size of this radius changes according to the number of vertices of the tree ($card(V)$):

$$r = min\{\gamma_{RRT^*} * (log(card(V)/card(V)) * 1/d, \delta\} \tag{5.4}$$

  Where d indicates the number of dimension of the configuration space, and δ is a pre-defined maximum value for the radius. The parameter γ_{RRT^*} is calculated according to the size of the configuration space $\mathcal{X}$ and its dimensionality.

- **MotCost**$(\mathbf{x_n}, \mathbf{x_m})$. This function calculates the cost of moving from the point x_n to the point x_m, according to the following equation:

$$MotCost(x_n, x_m) = \frac{c(x_n) + c(x_m)}{2} \|x_m - x_n\| \tag{5.5}$$

  Thus, the calculation takes into account the individual costs associated to each point (function $c(x)$) and the "distance" between them. Without loss of generality, we can assume that the cost function $c(x)$ for each point can be expressed as a weighted linear combination of a set of J feature functions $f_j(x)$ defining the task (similar to the cost (reward) function of the MDP that we learn in Chapter 3, or the one we use in Chapter 6):

$$c(x) = \sum_{j=1}^{J} \omega_j f_j(x) = \omega^T f(x) \tag{5.6}$$

- **Cost**$(\mathbf{x_n})$. Given a point $x_n \in V$, this function returns the accumulated cost of the path from the initial tree point x_{init} to the given tree point x_n:

$$Cost(x_n) = \begin{cases} c(x_n) & \text{if } x_n = x_{init} \\ Cost(x_{n-1}) + MotCost(x_{n-1}, x_n) & \text{otherwise} \end{cases} \tag{5.7}$$

- **Parent**$(\mathbf{x})$. Given a point $x \in V$ and $x \neq x_{init}$, this function return the point associated to the parent vertex of x.

Finally, the path is recovered from the minimum-cost leaf vertex in the goal region, and following its parent's vertices until the initial configuration. If a solution connecting the initial vertex with the goal region is not found, the closest path found to the goal region is returned. The path is then represented by a set of discrete configuration points $\zeta = \{x_1, x_2, \cdots, x_N\}$. Further mathematical details about asymptotic optimality an computational complexity can be found in (Karaman and Frazzoli, 2011).

5.4.2 GMM-RRT* planner

In the approach presented here the standard RRT* algorithm is extended with the learned GMM at two levels:

1. Including a new task-similarity cost into the evaluation of the node's costs. The GMM obtained encompasses the most likely configurations of the task. So, when a node is proposed to be added to the RRT* tree, a cost based on the GMM is derived and used. The objective is to increase the cost of those configurations that are unlikely according to the learned GMM. Thus, the likelihood is inverted to obtain that cost. Notice that the likelihood is a density of occurrence, so it is necessary to give a low bound to keep the inverse within the interval $[0, 1]$. To this end, it has been chosen to truncate the likelihood to a certain low value δ. Thus, cost function $c(x)$ expressed by (5.6), would take now the value of the inverted likelihood of the GMM in the point x.

2. Providing the planner with the most likely subspace to perform the sampling of the RRT*. If the RRT* knows the regions where the most likely paths are, then we can bias the sampling of the configuration space to these areas, reducing the probability of sampling useless states and, hence, reducing the computational costs to obtain a solution. To this regard, it is possible to define by a parameter the percentage of uniform sampling and GMM-based sampling to be used. In this way, the approach presented here can still take advantage of the random sampling. Therefore, we have modified the sampling procedure $SampleFree$ including new information: $SampleFree(x_{goal}, \epsilon, \mathbf{GMM}, \rho)$. Where with a probability ρ the sample is drawn from the regions of the configuration space covered by the Gaussian models (and obstacle-free) and leaving the rest to the uniform sampling of the configuration space.

These changes adopted still guarantee the probabilistic completeness of the modified RRT* planner proposed. Thus, although the sampling method is now

based on a GMM, it also keeps some percentage of uniform sampling that guarantees a feasible path if the execution time is enough. So, a good balance between the percentage of GMM-based sampling and the execution time given to the RRT* to plan needs to be negotiated.

5.5 Experimental results

In this section, we evaluate the proposed approximation against an existing framework (Claassens, 2010) in the two tasks considered. The data collection and the metrics employed are shown. We also highlight two important aspects about the suitability of both frameworks for planning: the management of the homotopies when solving the task and the generalization to deal with situations far away from the learned examples.

5.5.1 Data collection and unsupervised characterization

Several experiments were performed to retrieve exemplary trajectories to feed the GMM learning phase. Those experiments took place inside a laboratory of the Pablo de Olavide University. The robot was teleoperated by a person used to pilot it (an expert) while performing the scenes described in Section 5.2. The room was clear and free of obstacles between the person and the robot, and the study was recorded using a motion capture system (OptiTrack[2]) which provides the robot and people's poses automatically. An example picture of the laboratory can be seen in Fig. 5.4), in which the space for operation, the robot, and the hats employed for human detection are shown.

Then, the features $[d, \theta]$ were easily derived from the data, and the corresponding GMM models were obtained using the method described in Section 5.3. Those models are then integrated into the RRT* as explained in Section 5.4. In the execution of the task, the features are obtained from the people in the scene and a set of random samples are drawn from the GMM models. Then, these samples are transformed from the feature space to coordinates in the Cartesian space in the frame of the robot and used by the RRT* planner. This method is called GMM-RRT*. A set of GMM models in Cartesian space $[x, y]$ have also been derived to be used in the approach GMM-RRT presented in (Claassens, 2010), which will be used to baseline for comparison.

Figure 5.5 shows the recorded trajectories for the task of approaching a person. This task is also split into two subtasks depending on the relative orientation between the person and the robot. The case of robot and person facing each other is shown in the left image, while the case of the person placed backward

[2] `http://optitrack.com/motion-capture-robotics/`

Figure 5.4: A example view of the general set up of the laboratory for data-collection experiments.

to the robot is shown on the right. The GMM models learned for these subtasks are then combined to fulfill the general task, as shown in the down image of the figure, in which the models learned are depicted in the feature space.

5.5.2 Metrics

We propose two metrics to compare the obtained paths from the different planners and concerning the demonstrated trajectories.

The first metric is called *Trajectory Difference Metric* (TDM), which is defined as follows:

$$TDM(\zeta_D, \zeta_P) = \frac{1}{|\zeta_D|} \sum_{i=1}^{|\zeta_D|} \min_j \|\zeta_D(i) - \zeta_P(j)\| \tag{5.8}$$

where we sum the distances of every point i of the trajectory ζ_D to its closest point j of the trajectory ζ_P. Then we divide between to number of samples of the trajectory ζ_D, indicated by $|\zeta_D|$, in order to obtain an average distance. This metric gives an idea of the similarity of two given trajectories. The final metric is given by the averaged value of this metric for all the planned and demonstrated trajectories:

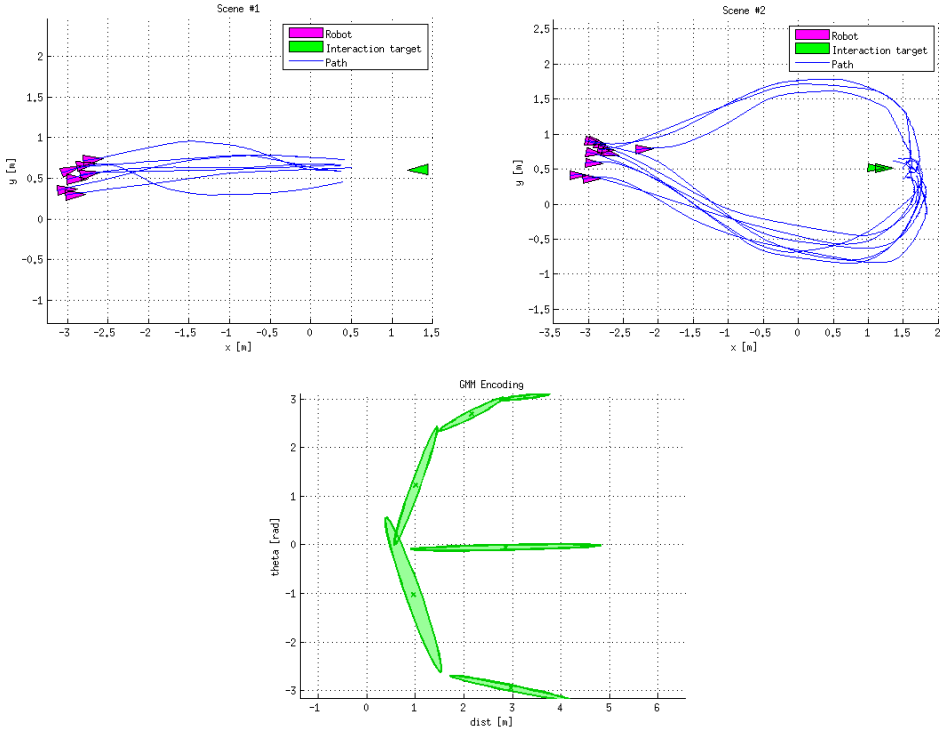


Figure 5.5: Trajectories collected and GMMs learned for the task of approaching a person. Up left: person and robot facing each other. Up right: person placed backwards to the robot. Down: Respective GMM models learned in the feature space $[d, \theta]$.

$$TDM = \frac{1}{|D|}\frac{1}{|P|}\sum_{D,P} TDM(\zeta_D, \zeta_P) \tag{5.9}$$

where D and P are the numbers of demonstrated and planned trajectories, respectively.

The second metric is the resulting averaged trajectory length ratio l_e, expressed as the mean of the ratio of the absolute value of the difference between the planned trajectories and the demonstrated trajectories lengths divided by the demonstrated trajectories' length:

$$l_e = \frac{1}{|D|}\frac{1}{|P|}\sum_{D,P} \frac{|l(\zeta_D) - l_(\zeta_P)|}{l(\zeta_D)} \tag{5.10}$$

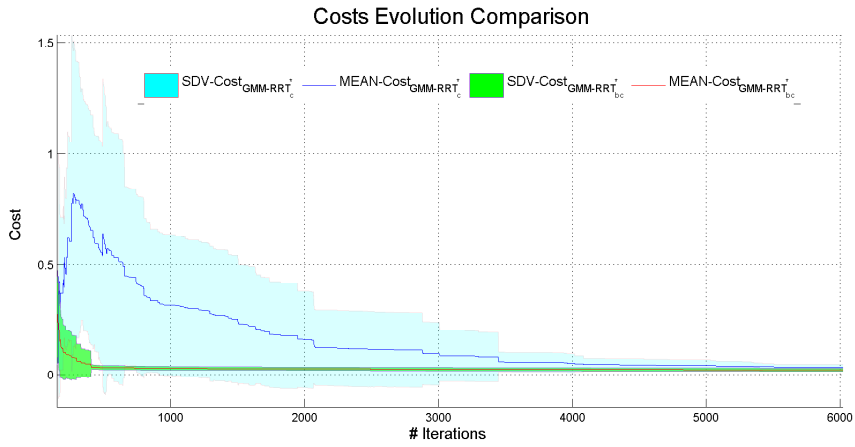


Figure 5.6: Evolution of the mean path cost value and the standard deviation according to the number of iterations executed. The GMM-RRT* approach with a 100 % of GMM sampling bias is represented by a red line for the cost, and a green area for its standard deviation. The GMM-RRT* approach without GMM sampling bias is shown with a blue line for the cost, and a light blue area for the standard deviation.

5.5.3 Results

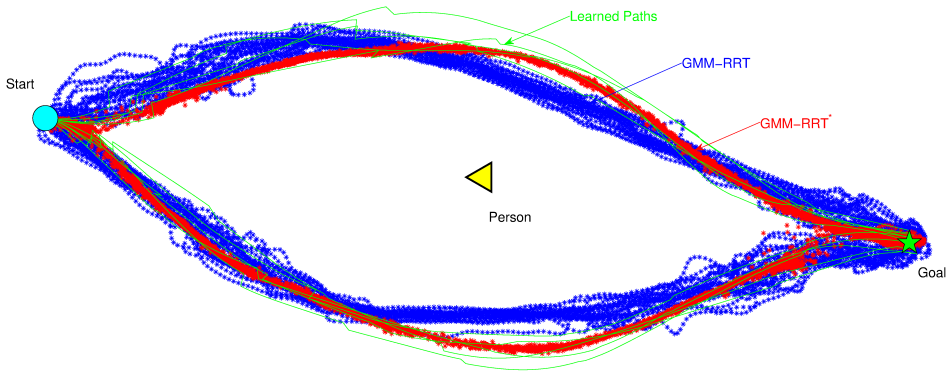
5.5.3.1 Path cost evaluation

Here we evaluate the convergence speed of the RRT*-based planner to the optimal path regarding costs. To do that, we employ the RRT* planner using a GMM-based cost and varying the percentage of sampling bias from the GMM models. Figure 5.6 shows the evolution of the solution path cost versus the number of iterations using 100% and 0% GMM bias, for the "Avoiding a Person" task. The allowed planning time for this comparison was 100 seconds to be sure to converge to the optimal cost.

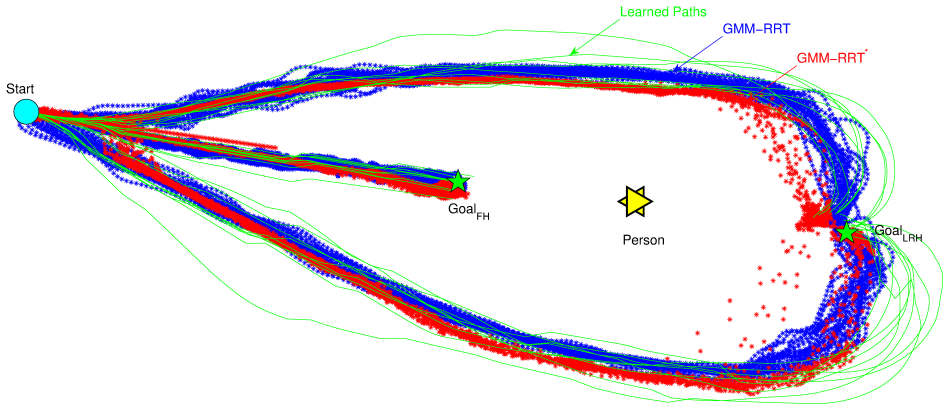
As can be seen in this example, the planner that includes the GMM sampling bias can converge to an optimal (or nearly optimal) path much faster than the version without GMM sampling bias. This result states the value of the bias and the need of balance the percentage of sampling from the GMM models employed.

5.5.3.2 Metrics performance

Here, we compare the proposed approach (GMM-RRT*) with a related approach of the state-of-the-art (GMM-RRT) (Claassens, 2010). This baseline approach for trajectory imitation also employs GMMs for statistical characterization of a



(a) Task 1: avoid a person



(b) Task 2: approach a person

Figure 5.7: Demonstrated and planned trajectories of the GMM-based planners. The "start" and "goal" states are shown, while the person's position and gaze is represented by a triangle. The demonstration paths are shown in green lines. The paths of the baseline planner are shown in blue, and the paths of the GMM-RRT* planner proposed in red. (5.7a) Avoiding a person task. (5.7b) Approaching a person task.

Table 5.1: Trajectory quality for both tasks. Smaller values are better for all metrics. The best values are highlighted in boldface.

	Task 1: Avoiding			**Task 2: Approaching**	
	TDM (m)	$l_e(\%)$		TDM (m)	$l_e(\%)$
Rigth Homotopy					
GMM-RRT*	**0.057** $\pm$ 0.015	**3.66** $\pm$ 2.64		**0.091** $\pm$ 0.018	**3.69** $\pm$ 2.54
GMM-RRT	0.080 $\pm$ 0.021	4.00 $\pm$ 2.51		0.092 $\pm$ 0.016	4.96 $\pm$ 3.45
Left Homotopy					
GMM-RRT*	**0.048** $\pm$ 0.011	4.96 $\pm$ 2.37		**0.061** $\pm$ 0.029	**4.22** $\pm$ 2.50
GMM-RRT	0.068 $\pm$ 0.017	**4.31** $\pm$ 3.1		0.067 $\pm$ 0.017	4.56 $\pm$ 3.62
Frontal Homotopy					
GMM-RRT*				0.038 $\pm$ 0.014	11.35 $\pm$ 5.35
GMM-RRT				**0.034** $\pm$ 0.018	**8.55** $\pm$ 3.34

set of demonstration trajectories. Then, unlike our approach, the search tree of an RRT planner is completely kept within the statistical model learned (100 % of sampling bias of the GMM model). Moreover, the sampling process is constrained so that the samples can be drawn only from following modes of the GMM learned. Further details of the approach can be consulted in (Claassens, 2010).

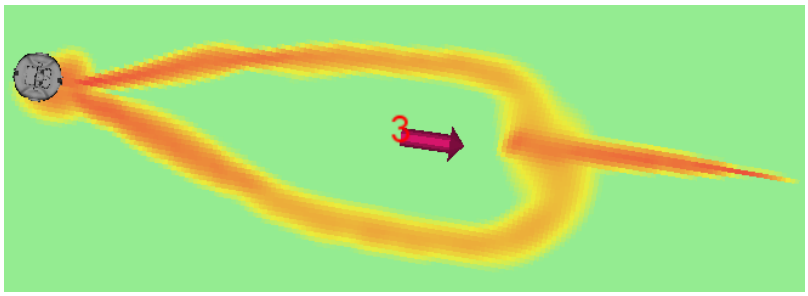
For the comparison, the following mixed-sampling strategy was adopted in our approach: the 95% of the samples were drawn from the learned GMM while the remaining 5% of the samples were uniformly obtained. We aim to take advantage of the learned models and also allow a degree of randomness when sampling configurations. This feature is only applicable to the proposed GMM-RRT* planner. In the case of GMM-RRT planner, the commented constraints in the construction of the RRT tree do not allow uniform sampling of the space.

Figure 5.7 shows graphically a set of 25 paths obtained by each planner for both tasks proposed. The different valid homotopies are also taken into account. Visually, both planners seem to mimic quite well the demonstrated paths followed in the tasks.

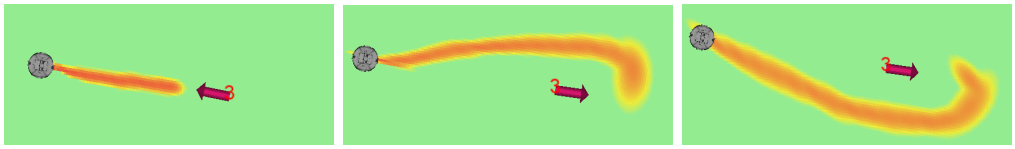
To have a clearer comparison of the planners' ability to imitate the demonstrated trajectories, we make use of the metrics presented in Section 5.5.2. Table 5.1 shows the values of the metrics obtained when the different planners are used in both tasks. As can be observed, the planner proposed in this work outperforms in nearly every homotopy considered.

5.5.3.3 Homotopy management

As commented in Section 5.2, the tasks being learned by the robot may be composed by several homotopies that are equally valid for reaching the goal. For instance, Fig. 5.8 illustrates the compound model of the Approaching-a-Person task. As can be seen, the compound model covers the subtasks of approaching the person from the front or the back. Moreover, this latter can be reached by two homotopies, surrounding the person from the left or right side.



(a) Compound model for the Approaching-a-person task



(b) Three different GMM models for fulfilling the Approaching task

Figure 5.8: Models learned for the Approaching-a-person task. (5.8a) compound model of the three different GMMs. (5.8b) Separate GMM models: front approach, left homotopy for back approaching, and right homotopy for back approaching respectively.

This is a clear advantage of the GMM-RRT* planner proposed over the baseline. In case of the GMM-RRT planner, not only the social situation has to be known beforehand to choose between the homotopy models, but also once a model has been selected, an obstacle may hamper the execution of the plan. In such a situation, both planners can take advantage of the variability in the execution of the demonstrated task, encoded by the GMMs covariance matrices, to avoid the obstacle, but sometimes this could not be enough. An example of this situation is illustrated in Fig. 5.9. There, it can be observed how the GMM-RRT planner is not able to find a free path to the goal, although it exists. However, the ability of the proposed GMM-RRT* planner to manage different homotopies allows it to choose between the possible homotopies to reach the goal naturally.

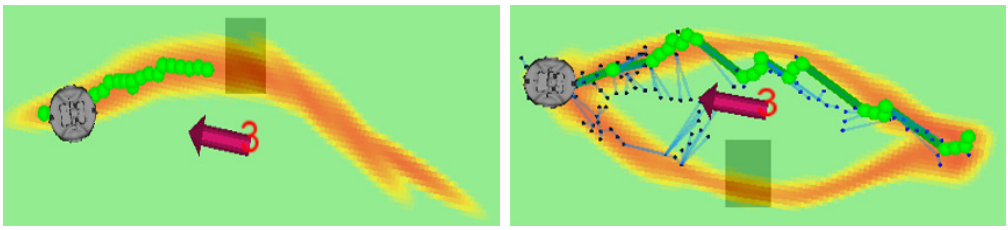


Figure 5.9: Avoiding-a-person task. Shadowed area represents an obstacle. Left: GMM-RRT planner can deal with only one homotopy, leading to a block situation. Right: GMM-RRT* planner managing a two-homotopies model, and thus, reaching correctly the goal.

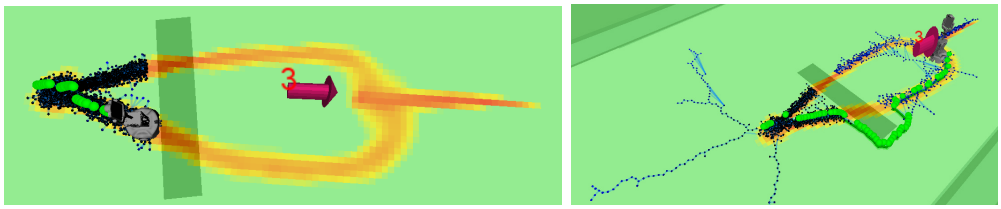


Figure 5.10: Mixed-sampling strategy for generalization of the GMM-RRT* approach. Left: Simulated trial without uniform sampling. Right: 95% of GMM sampling bias and 5% of uniform sampling.

5.5.3.4 Generalization

Another important aspect to be evaluated is the generalization to other scenarios different from those used in the learning. The GMM models are very good in mimicking the situations employed to build them. However, robot navigation tasks involve several different scenarios and situations that can also change dynamically, and the GMMs can hardly deal with them. Nevertheless, the combination of the GMMs with the RRT* allows us to take advantage of the ability of the RRT* planners to overcome those situations by exploration the state space.

The Approaching-a-person task is modified in Fig. 5.10 so that an obstacle (as a rectangular shadowed area) is introduced in the scenario blocking the two possible robot path to the goal encoded in the GMMs learned. In that situation, the planner is not able to find a path to goal if only samples from the GMM model can be drawn (left image). On the other hand, the GMM-RRT* planner still can reach the goal is a mixed-sampling strategy is adopted (right image): the percentage of uniform sampling allows to randomly explore the state space outside the learned space encoded through the GMMs, and therefore a path to the goal avoiding the obstacle can be found.

The simulations showed in Fig. 5.10 were within a 10 seconds time horizon.

Depending on the task to perform, the percentage of bias, thus, offers a trade-off between the imitation capabilities and the planning time required to obtain a path. An adaptive solution that dynamically modifies this bias after obtaining a first good path is left for future work.

5.6 Conclusions

In this chapter, we presented a planning algorithm based on RRT*, which is capable of inferring the most suitable socially-aware paths in two particular situations (or tasks): avoiding a standing person and approaching forwards or backward a standing person. Both tasks are statistically characterized from real experiments by using two different GMMs, which are later used in two ways: to guide the state-space sampling step and to include a cost term into the standard cost function employed in the RRT* algorithm.

Unlike discrete MDP-based approaches, as seen in Chapter 3, the use of the RRT* planner allows us to work in continuous spaces. It also scales better with larger state spaces and higher dimensionality. Additionally, the statistical characterization of the demonstration trajectories based on GMMs allows us to lead the planner to mimic the provided behavior correctly.

We evaluated the approach presented here jointly with a state-of-the-art algorithm based on RRT and GMM. The comparison included a metric performance to measure the similarity between the planned trajectories with the learned ones, and how both approaches performed in simulated scenarios that can include slight differences from where they were learned. Although the RRT based planner was quicker to get a suitable path, the approach presented here was able to manage homotopies. It also generalized better when tackling with unexpected situations (such as obstacles), thanks to the trade-off between the uniform and the biased sampling of the configuration space.

However, the proposed approach presented some drawbacks. First, we have to define the set of features that describe the task to be imitated. Nevertheless, in complex problems, like human-aware navigation, it is not easy to determine the key features that better describe the task considered. Second, some parameters need to be manually specified as the number of modes of the GMM model to be learned. And finally, to consider the possible obstacles in the modeling is complex. In our approach, they are not considered in the GMM modeling but we used the RRT* to deal with them. Unfortunately, this can not be enough to overcome all the possible situations that can arise.

In any case, we think the approach can be satisfactorily employed in specific situations, like the task of approaching a human target in a socially-acceptable manner, as we implemented in the TERESA project (Chapter 7). Further-

more, to overcome some the drawbacks commented, a different Learning-from-Demonstration approach for the general human-aware navigation problem using RRT* planners is presented in the next Chapter.

Chapter 6

Learning robot navigation behaviors by demonstration using a RRT* planner

"May the Force be with you."
Star Wars, 1977

This chapter presents an approach for learning navigation behaviors from demonstrations using Optimal Rapidly-exploring Random Trees (RRT*) as the main planner. A new learning algorithm combining both Inverse Reinforcement Learning techniques (IRL) and RRT* planners is developed to learn from expert's demonstrations the RRT*'s cost function that leads the planner to behave similarly to the demonstrated examples.

In Chapter 3, IRL based on discrete MDPs was employed to learn a simple reward function for reactive motion planning. To overcome some of the drawbacks found, we decide on taking advantage of a more flexible path planner in continuous spaces (RRT*), as we also introduced in Chapter 5. In contrast, here we present a new approximation that combines the RRT* planner with IRL concepts to induce social behavior into it. A comparison with other state-of-the-art algorithms shows how the method can recover the underlying behavior demonstrated by the expert.

6.1 Introduction

As commented in previous chapters, IRL is a successful technique to learn human behaviors from demonstration. In most of the existing models, the IRL technique makes use of MDPs as the underlying process (Henry et al., 2010; Ramón-Vigo et al., 2014; Vasquez et al., 2014). MDPs present some disadvantages for robot motion planning, like poor scalability with the involved large (and typically continuous) state spaces, which leads to high computational complexity.

Different solutions have been considered to deal with the main MDP drawbacks. The computational cost is managed in (Michini et al., 2013) by using a

Bayesian nonparametric mixture model to divide the observations and obtain a group of simpler reward functions to learn. In (Levine and Koltun, 2012), deterministic MDPs with large, continuous state and action spaces are handled by considering only the shape of the learned reward function in the neighborhood of the experts demonstrations. Furthermore, neural networks are applied in (Xia and Kamel, 2016) to learn non-linear policies. In (Okal and Arras, 2016a), a socially normative behavior is learned using flexible graph-based representation to model the underlying MDP.

Other authors try to replace or represent the MDP by another process according to the aim of the task. In the called Maximum Margin Planning (MMP) method (Ratliff et al., 2006), the MDP can be replaced by an A* search algorithm. Also in (Shiarlis et al., 2017a) the MMP method is extended to learn RRT* cost functions instead of using an A* search algorithm. Different car driving styles are learned in (Kuderer et al., 2015) by modeling the car dynamics using a time-continuous spline representation instead of an MDP. Another example is (Kretzschmar et al., 2016), where the cooperative navigation behavior of humans is learned using Hamiltonian Markov chain Monte Carlo sampling to compute the feature expectations over high-dimensional continuous distributions.

We present here a novel approach to learn navigation behaviors from demonstrations, which we have called Rapidly-exploring random Trees Inverse Reinforcement Learning (RTIRL) (Pérez-Higueras et al., 2016a, 2017). We make use of IRL concepts (Ng and Russell, 2000) and sampling-based costmap planners, like Optimal-RRTs (RRT*)(Karaman and Frazzoli, 2011), to identify the RRT cost function that best fits the example trajectories. In Chapter 3, we present an approach with a gross discretization and a low dimensionality of the state and feature space to keep the MDP computationally tractable. In contrast, here we make use of RRT* path planners instead of MDPs, which are quite widespread planners in the robotics community. The method exploits the advantages of the RRT* planners to deal with continuous state and control spaces and to scale with the state size. It is also computationally faster and general enough to be applied in different scenarios.

In particular, we consider the application for semi-autonomous telepresence robot developed in the European project TERESA (Shiarlis et al., 2015), as we did in Chapter 5. Our goal is to increase the autonomy of such robots, freeing the users from the low-level navigation tasks. In this setup, it is very natural to obtain navigation data from the user, considering them as examples and using them to learn social navigation behaviors. In particular, we employ navigation demonstrations in static scenarios (non-moving pedestrians), that we use later in dynamic scenarios by replanning at high frequency.

The proposed approximation is compared against other similar approaches from state of the art. The results show how the method adequately recovers the demonstrated behavior.

6.2 Learning a RRT* cost function from demonstrations

6.2.1 IRL formulation with RRT* planners

RRT* is a technique for optimal motion planning employed extensively in robotics (Karaman and Frazzoli, 2011). It is flexible and easily adapted to different scenarios and problems. RRT* approaches reason about collisions with obstacles at moderate computational cost even in high dimensionality. They explore the configuration space to obtain optimal paths on cost spaces, and the kinodynamic extension allows reasoning about the robot dynamics.

The RRT* algorithm is presented and explained in detail in Chapter 5. According to Eq. (5.6), the cost of a point x of the configuration space, can be calculated as a weighted linear combination of a set of features $f(x) = [f_1(x), f_2(x), \cdots, f_J(x)]^T$. The particular features will depend on the task to be learned, and are used to extract the information from the robot configurations that are relevant for that particular task. While not explicitly indicated, the value of the features will also depend on the scenario s (given by the position of obstacles, people, goals and any other relevant information external to the robot) where the planning takes place.

The cost of a path ζ is then the sum of the cost for all points in it. Particularly, in the RRT*, the cost of a path is the sum of the sub-costs of moving between pairs of points in the path, according to Eq. (5.5) presented in the previous chapter:

$$C(\zeta) = \sum_{i=1}^{N-1} \frac{c(x_i) + c(x_{i+1})}{2} \|x_{i+1} - x_i\| \tag{6.1}$$

$$= \omega^T \underbrace{\sum_{i=1}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2} \|x_{i+1} - x_i\|}_{f(\zeta)} \tag{6.2}$$

$$= \omega^T f(\zeta) \tag{6.3}$$

where $f(\zeta)$ is called the feature count vector of path ζ. Thus, for a given weights vector ω, the algorithm will return trajectories that try to minimize this cost.

Given a set of demonstration trajectories by an expert,

$$\mathcal{D} = \{\zeta_1, \zeta_2, \cdots, \zeta_D\} \tag{6.4}$$

and the definition of the cost function $c(x)$, given by Eq. (5.6), the problem of learning from demonstrations, in this setup, means to determine the weights ω that lead our planner to behave similarly to these demonstrations.

The concept of similarity in this context is ambiguous. One may want to reproduce the same trajectories in the same situations. However, it is necessary to learn a representation that can generalize to other cases. As in (Abbeel and Ng, 2004; Kuderer et al., 2015), this can be achieved by using the mentioned features as a measure of similarity. The objective is then to model the underlying trajectory distribution of the expert $p(\zeta|\omega)$ with the constraint that the expected value of the features for the path generated by the model is the same as the expected value of the features for the given demonstrated trajectories:

$$\mathbb{E}(f(\zeta)) = \frac{1}{D}\sum_{i=1}^{D} f(\zeta_i) \tag{6.5}$$

There are many distributions $p(\zeta|\omega)$ that achieve the previous constraint. Applying the Maximum Entropy Principle (Ziebart et al., 2008; Kretzschmar et al., 2014) to the IRL problem leads to the following form for the probability density for the trajectories returned by the demonstrator:

$$p(\zeta|\omega) = \frac{1}{Z(\omega)} e^{-\omega^T f(\zeta)} \tag{6.6}$$

where the partition function $Z(\omega) = \int e^{-\omega^T f(\zeta)} d\zeta$ is a normalization function that does not depend on ζ. Thus, this model assumes that the expert is exponentially more likely to chose a trajectory with lower cost $\omega^T f(\zeta)$.

Under this model, we can estimate the log-likelihood of a particular trajectory ζ:

$$\mathcal{L}(\zeta|\omega) = log \frac{1}{Z(\omega)} e^{-\omega^T f(\zeta)} = \tag{6.7}$$

$$-logZ(\omega) + (-\omega^T f(\zeta)) \tag{6.8}$$

Therefore, assuming conditional independence of the demonstrations, the (log-)likelihood of the set of demonstrated trajectories is given by:

$$\mathcal{L}(\mathcal{D}|\omega) = -D\log(Z(\omega)) + \sum_{i=1}^{D}(-\omega^T f(\zeta_i)) \tag{6.9}$$

The gradient of this log-likelihood with respect to ω is given by:

$$\frac{\partial \mathcal{L}(\mathcal{D}|\omega)}{\partial \omega} = D \underbrace{\frac{1}{Z(\omega)} \int f(\zeta) e^{-\omega^T f(\zeta)} d\zeta}_{\mathbb{E}(f(\zeta))} - \sum_{i=1}^{D} f(\zeta_i) \tag{6.10}$$

$$= D(\mathbb{E}(f(\zeta)) - \frac{1}{D} \sum_{i=1}^{D} f(\zeta_i)) \tag{6.11}$$

We arrive at the primary constraint in (6.5) by setting this gradient to zero. That means, as indicated in (Kuderer et al., 2012), that the IRL problem under the Maximum Entropy distribution is equivalent to maximizing the likelihood of the demonstrations when assuming an exponential distribution.

Maximizing the likelihood, Eq. (6.9), to obtain the set of weights cannot be solved analytically. But the gradient in (6.11) can be used to apply gradient ascend techniques to solve this problem. As mentioned in (Kuderer et al., 2015), this gradient can be intuitively explained. If the value of one of the features for the trajectories returned by the planner is higher than the value in the demonstrated trajectories, the corresponding weight should be increased to penalize those trajectories.

The main problem with the computation of the previous gradient is that it requires to estimate the expected value of the features $\mathbb{E}(f(\zeta))$ for the generative distribution (6.6) over the continuous space of trajectories.

$$\mathbb{E}(f(\zeta)) = \frac{1}{Z(\omega)} \int f(\zeta) e^{-\omega^T f(\zeta)} d\zeta \tag{6.12}$$

In (Kretzschmar et al., 2014), a probabilistic generative model for trajectories is derived from data, and the expectation is computed by Monte Carlo Chain sampling methods, which are very computationally demanding. One option is to approximate the previous expectation by the features of the most likely trajectory (Kuderer et al., 2015). In our case, we will approximate the expert by the RRT* planner onboard the robot. Being an asymptotically optimal planner, for some given weights, the RRT* will provide the trajectory that minimizes the cost (the most likely) given infinite time. As the planning time is limited, the RRT* will provide trajectories with some variability on the features, and thus the feature expectation $\mathbb{E}(f(\zeta))$ is computed by running several times the planner between the start and goal configurations. That is then used to calculate the gradient and adapt the weights employed in the RRT* planner.

The experimental results will show that the method can recover the behaviors taught by the expert.

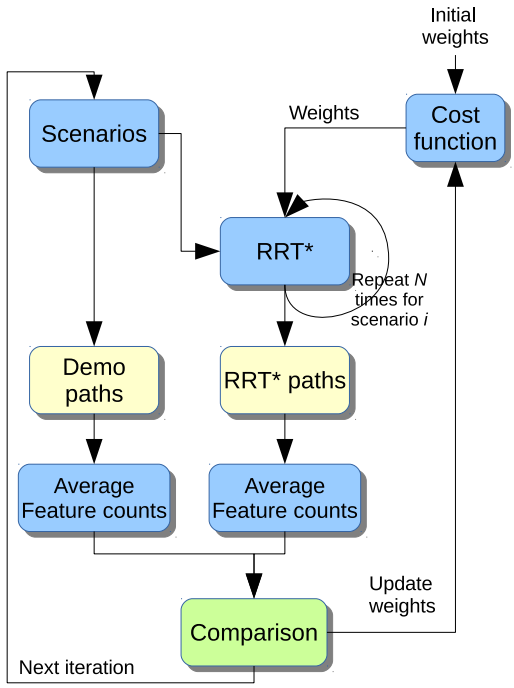


Figure 6.1: General learning scheme proposed to adjust the weights of the cost function of a RRT* planner.

6.2.2 RTIRL algorithm

The general scheme of the learning algorithm proposed is presented in Figure 6.1. First, we define a scenario as a particular situation where to perform the task proposed (for instance, a particular position of the obstacles, people positions, and orientations, starting and goal positions, for the task of social navigation). So, for each different scenario $s \in \mathcal{S}$ present in the demonstrations, we plan a path to the respective goal using the RRT* planner. The planner has a limited time to plan, so small variations of the resulting path can arise. So, as commented before, we deal with that by repeating the plan some times and calculating the average values over the number of repetitions. Then, the feature count of the demonstration paths and the RRT* paths are calculated and averaged over all the scenarios. The difference between these values is used to update the weights of the cost function at each iteration. Thus, we get the weights that make the cost function fit better the demonstration paths as the output of the algorithm.

The approximation is described in more detail in Algorithm 6.1. First, in Line 1 we obtain the average feature count $\overline{f}_{\mathcal{D}} = \frac{1}{D}\sum_{i=1}^{D} f(\zeta_i)$ from the example

Algorithm 6.1: RTIRL

Input: Trajectory examples $\mathcal{D} = \{\zeta_1^1, \ldots, \zeta_D^S\}$ in $\mathcal{S}$ scenarios
Output: Function features weights $\omega = [\omega_1, \ldots, \omega_J]^T$

1: $\overline{f}_{\mathcal{D}} \leftarrow calculeAvgFeatureCounts(\mathcal{D})$
2: $\omega \leftarrow randomInit()$
3: **repeat**
4: **for each** $s \in \mathcal{S}$ **do**
5: **for** $rrt\_repetitions$ **do**
6: $\zeta_i \leftarrow getRRTstarPath(s, \omega)$
7: $f(\zeta_i) \leftarrow calculeFeatureCounts(\zeta_i)$
8: **end for**
9: $\overline{f}^s_{RRT^*} \leftarrow (\sum_{i=1}^{rrt\_repetitions} f(\zeta_i))/rrt\_repetitions$
10: **end for**
11: $\overline{f}_{RRT*} \leftarrow (\sum_{i=1}^{S} \overline{f}^i_{RRT^*})/S$
12: $\nabla\mathcal{L} \leftarrow (\overline{f}_{RRT*} - \overline{f}_{\mathcal{D}})$
13: $\omega \leftarrow UpdateWeights(\nabla\mathcal{L})$
14: **until** convergence
15: **return** ω

trajectories $\mathcal{D}$ in $\mathcal{S}$ different scenarios. The feature counts are obtained as the addition of the feature values of pairs of nodes of the trajectory evaluated using (6.2).

Then we initialize the weights with a random value (Line 2). It is noteworthy that the weights are being normalized during the learning iterations, and the features values for each node are also normalized for the sake of clarity of representation, but this is not a requirement of the algorithm.

The key point is the gradient given by (6.11), which requires a comparison of the features counts obtained from the example trajectories and the expected value from the RRT* planner. The latter is obtained by running $rrt\_repetitions$ times the planner for the current weight values for each scenario considered (Line 6), and estimating and normalizing the features counts (Line 7). In Lines 9 and 11, the averaged values are computed.

Based on this comparison the weights of the cost function are updated using gradient descent (line 13):

$$\omega_{i+1} \leftarrow \omega_i - \frac{\lambda}{\phi} * \nabla\mathcal{L}_i \tag{6.13}$$

where ϕ is a value for stabilization which is being incremented after each iteration, λ is an adjusting factor of the equation and $\nabla\mathcal{L}_i = \frac{\partial\mathcal{L}(\mathcal{D}|\omega)}{\partial\omega_i}$ is the i-th component of the gradient.

Finally, the learning process finishes when the variations of the weight values keep under a certain convergence value ϵ.

6.3 Features for social navigation

The general learning approach presented above can be applied to any task that can be solved with a RRT* planner and by defining the adequate features that describe the task properly. From this section on, we apply the algorithm to learn the particular task of human-aware navigation. So, a set of features specially designed for that task is presented here.

The social navigation task considered here involves the robot navigation in different house environments like rooms and corridors where some persons stand in various positions, the robot has to avoid them to reach the goal.

The selection of the adequate features involved in the social navigation task is not trivial (Ramón-Vigo et al., 2015). A review and analysis of the most used features in the literature are presented by Vasquez et al. (2014). A set of commonly-used features have been considered here by taking into account these previous works. In particular, they are the distance to the goal, the distance to the closest obstacle, and the distance from the robot to the people in the scene and the angle of the robot position concerning the people α, as depicted in Figure 6.2. We are not using dynamic features as velocities or accelerations here since we are considering learning from static scenarios, where the pedestrians are still. Then, the learned behavior is applied to dynamic scenarios by periodic re-planning at high frequency to capture the new situations in the environment. The adaptation of the learning algorithm to employ demonstrations in dynamic scenarios is left for future work.

Thus, five functions based on the features presented are combined to obtain the cost function employed in the RRT* planner. The functions are computed for each sample x_k of the configuration space. The first one is just the computation of the Euclidean distance from the robot position to the goal:

$$f_1(x_k) = \|x_k, x_{goal}\| \tag{6.14}$$

The second feature function uses the distance to the closest obstacle for each node x_k, with the aim of motivating the robot to keep some distance from the obstacles. This value grows when the distance to the obstacle decreases. A distance transform function is employed to obtain the distance to the closest obstacle for each sample (Borgefors, 1986). Then, a normalized value is obtained by using this distance according to Eq. (6.15).

$$f_2(x_k) = \frac{a_1}{\gamma * (\|x_k, x_{closest\_obs\_k}\| + a_2)} \tag{6.15}$$

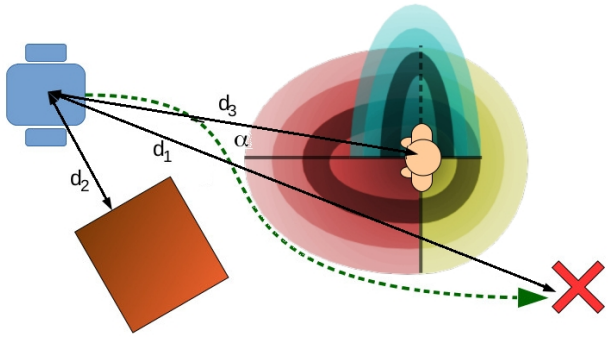


Figure 6.2: Some of the features employed in the social cost function learned. $d1$, distance to the goal. d_2, distance to the closest obstacle. d_3, distance from the people to the robot. α, angle between the person front and the robot location. The three Gaussian functions deployed over the people are also represented.

where a_1 and a_2 are the parameters that define the function and take the values 2 and 0.2 respectively. γ is a normalization parameter that takes the value 10 in this case.

Then, other three feature functions representing a proxemics cost concerning the people in the environment are employed. These cost functions are defined by Gaussian functions and are inspired by the model used by Kirby et al. (2009). One Gaussian function is placed in the front of the person, another one in the back, and the last one in the right side of the person. The shape of these Gaussian functions can be seen in Fig. 6.2. The first Gaussian function is asymmetric and placed in the front of the person with $\sigma_h = 1.20$ (m) being the variance in the direction the person is facing, and considering a smaller variance towards the sides $\sigma_s = \sigma_h/1.5$. The second Gaussian is placed in the back of the person with $\sigma_h = \sigma_s = 0.8$. Finally, a third Gaussian is placed in the right side of the person with $\sigma_h = 0.8$ and $\sigma_s = 0.8/2.5$. This function can be useful in corridor scenarios where, usually in Europe, the people usually walk on the right-hand side (Kruse et al., 2013). Another one could be placed on the left side, so the system should learn the correct weight for each side.

These cost functions p depend on the distance (d_{jk}) and relative angle (α_{jk}) of the robot position x_k with respect to each person j in the scenario. The costs due to all persons in the scenario are integrated according to the next expressions, where $\mathcal{P}$ represents the set of people:

$$f_3(x_k) = max\{p_{front}(d_{jk}, \alpha_{jk}), \forall j \in \mathcal{P}\} \tag{6.16}$$

$$f_4(x_k) = max\{p_{back}(d_{jk}, \alpha_{jk}), \forall j \in \mathcal{P}\} \tag{6.17}$$

$$f_5(x_k) = max\{p_{right\_side}(d_{jk}, \alpha_{jk}), \forall j \in \mathcal{P}\} \tag{6.18}$$

The amplitude of these two-dimensional Gaussian functions is set to 1, so the values returned by the cost functions p are in the range $[0, 1]$. Moreover, as we are planning in fixed-size area, the two feature functions based on distance are also normalized by dividing by the maximum distance possible between two points in the area. Thus, the values of the J (5 in this case) feature functions are normalized and the cost function for each node x_k is built adding its weighted values $c(x_k) = \sum_{i=1}^{J} \omega_i f_i(x_k)$ where $\omega_i \in [0, 1]$ and $\sum_i \omega_i = 1$.

Finally, the total cost of the N nodes of the path ζ is obtained based on the *motion-cost* function employed by the RRT* algorithm to calculate the cost of moving from one node to the next one according to Equation (6.1).

6.4 Algorithm validation

In this section, we validate the ability of the proposed RTIRL algorithm to recover the cost function and the behavior implicitly encoded by the demonstrations. To do so, a ground-truth cost function is employed to obtain a set of demonstration trajectories from a motion planner optimizing such function. Then, these trajectories are used by the proposed learning algorithm. This way, the learned cost function can be directly compared with the ground-truth one.

Moreover, we present a comparison with two learning from demonstration algorithms of the state-of-the-art: the Maximum Margin Planning algorithm (MMP) (Ratliff et al., 2006), which uses an A* algorithm as a planner, and a new learning algorithm called Rapidly Exploring Learning Trees (RLT*) (Shiarlis et al., 2017a), which adapts MMP to use a RRT* as a planner. These algorithms have been chosen because they follow the same idea of using a planning algorithm instead of an MDP, and these planning algorithms are also two of the most used planners in robot navigation. The implementation code of the RTIRL and RLT* learning algorithms is publicly available under BSD license in the Github of the Service Robotics Lab of the Pablo de Olavide University[1].

The MMP is a method for learning to plan which attempts to automate the mapping from perception features to costs (or rewards) as well as IRL tries to find the weights of a cost (or reward) function of an MDP. The idea behind MMP is to determine the set of weights that make that the cost of the demonstrated trajectories is lower than any other path between the same start and goal configurations. For better generalization, the cost function is augmented by a margin that reduces the costs of paths far from the demonstrations. The notion of closeness to demonstrations is defined by a loss-function $\mathcal{L} : \zeta \times \zeta_D \rightarrow \mathbb{R}_+$ which defines for each path ζ how much the learner pays for failing to match the example behavior ζ_D. This loss function value is inverted $(1 - \mathcal{L}(\zeta(k), \zeta_D))$ and

[1] `https://github.com/robotics-upo/upo_nav_irl`

added to the cost function. This way, the algorithm tries to make any demonstrated path better than any other path by a margin that scales with the size of the loss.

The loss-function employed in the A*-based MMP algorithm (Ratliff et al., 2006) counts the number of states that the planner visits but the teacher did not. It is also recommended to relax this function so that nearby paths are also admissible. In the case of RLT* the loss-function used takes into account the nearby paths but it is not explicitly specified (Shiarlis et al., 2017a). Therefore, in our implementation, we have followed the idea of admitting also nearby path by defining a loss-function, used in both methods (MMP and RLT*), which add a cost for each point of the evaluated path based on the Euclidean distance from that point to the closest point of the demonstrated path according to:

$$\mathcal{L}(\zeta(k), \zeta_D) = \begin{cases} 1 & \text{if } d_{min}(\zeta(k), \zeta_D) > 1 \\ d_{min}(\zeta(k), \zeta_D) & \text{otherwise} \end{cases} \tag{6.19}$$

where the function d_{min} calculates the Euclidean distance between the point k of the path ζ and the closest point of the demonstration path ζ_D.

6.4.1 Datasets for learning and testing

To validate the algorithm performance, we have used a set of demonstration trajectories obtained by a planner optimizing a given ground-truth cost function unknown to the learner. That is, the demonstration trajectories have been recorded by the planner using a cost function with the features explained in section 6.3 and a known set of weights. This way, we can compare the weights learned with the ground-truth weights and the path costs of the demonstrations and the learned paths. The paths generated with the cost function computed with the learning algorithm are expected to recover the behavior of the demonstration trajectories. Moreover, we should obtain approximately the same weights of the cost function used to generate the demonstration trajectories, our ground-truth, in case that a unique set of weights can replicate the desired behavior.

Particularly, we have employed the ground-truth cost function to record 30 trajectories, each one in a different demonstration scenario (different initial robot position, goal position and person positions in different parts or rooms in a house map). Also, we varied the number of persons in the scenes from 1 to 3. These demonstrations have been recorded by using the RRT* planner with the ground-truth cost function proposed. All the trajectories are performed in a fixed area of 100 m^2 with the robot in the center. And the time for plan calculation employed is 10 seconds, which have been seen that it is enough to converge close to the optimum given the space for planning.

Testing and validation of the approximation will be based on splitting the 30 different demonstration configurations into 20 trajectories to learn the weights

Table 6.1: Demonstration trials

Trial 1	Learning	[1-20]
	Evaluation	[21-30]
Trial 2	Learning	[11-30]
	Evaluation	[1-10]
Trial 3	Learning	[1-10, 21-30]
	Evaluation	[11-20]

of each cost function, and 10 configurations to compare the resultant paths. An example of the configurations employed in the validation can be seen in Figure 6.3 where the scenarios 9 and 19 are presented. To have small, well-defined sets of trajectories in different enough scenarios, we have grouped the trajectories used for learning and evaluation in three trials for cross-validation according to Table 6.1.

The time the RRT* is allowed to plan a path during the execution of the learning algorithms RLT* y RTIRL is 3 seconds (other times are also considered in section 6.4.7). In the case of the MMP learning, the resolution of the discretization of the state space for the A* planning is 0.05 m, which is considered a good resolution to get a smooth A* path. Finally, 100 iterations of each learning algorithm have been performed.

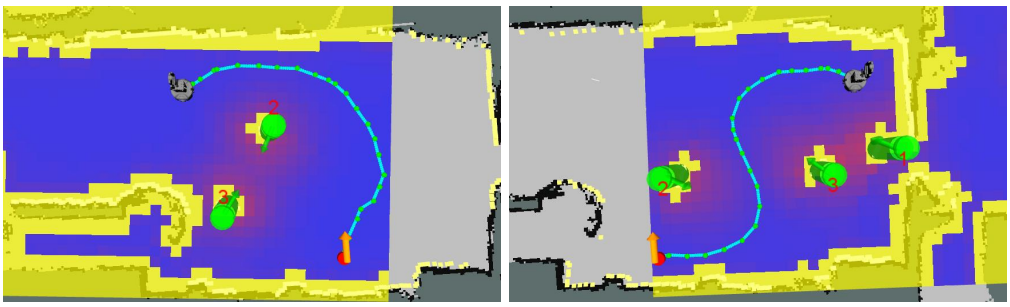


Figure 6.3: Some of the scenarios employed in the validation (scenario 9 and scenario 19 respectively). The green cylinders represent the people in the scene indicating their orientations with an arrow. A low-resolution coloured costmap is also shown.

6.4.2 Weights comparison

As the ground-truth weights are known (ω_{gt}), we can calculate the relative errors committed in the final weights learned according to:

$$RE_{\omega} = \frac{\|\omega_{gt} - \omega\|}{\|\omega_{gt}\|} \tag{6.20}$$

These relative errors obtained in the weights by each algorithm for the 3 trials are shown in Table 6.2. As can be seen, the RTIRL approximation can recover the ground-truth weights with a small error for all the trials committing a final average error much lower than the other approximations.

Table 6.2: Relative error in the weights committed by the learning algorithms in the three trials. The mean values and the standard errors are also shown.

RE_{ω}	RTIRL	RLT*	MMP
Trial 1	**0.101**	0.208	0.524
Trial 2	**0.057**	0.197	0.361
Trial 3	**0.047**	0.109	0.561
Mean:	$\mathbf{0.068 \pm 0.016}$	0.171 ± 0.031	0.482 ± 0.062

Once the learning phase has finished, we can compare the 10 trajectories reserved for evaluation of the 3 trials with the trajectories computed by the RRT* (and A* planner in case of MMP) with the estimated cost functions in order to visualize how close to the trajectories are in other scenarios. The planning time for the RRT* planner and the space discretization for the A* planner are the same that the ones employed in the learning phase. To overcome the randomness of the RRT* algorithm, 5 paths are planned, and an average computation over these paths is performed. The comparison is performed using the following metrics:

6.4.3 Path dissimilarity comparison

The first considered metric that we called dissimilarity is used to compare how different are two given paths in a Euclidean space. It is inspired by the metric for path comparison expressed in Eq. 5.8 of Chapter 5. This metric is an estimation of the area contained between two paths ζ_1 and ζ_2 by employing the Riemann's summation:

$$\begin{aligned} &dissimilarity(\zeta_1, \zeta_2) = \\ &\frac{\sum_{k=1}^{N-1} \frac{d(\zeta_1(k),\zeta_2)+d(\zeta_1(k+1),\zeta_2)}{2} \|\zeta_1(k) - \zeta_1(k+1)\|}{\delta} \end{aligned} \tag{6.21}$$

where the function d calculates the Euclidean distance between the point k of the path ζ_1 and its closest point of the path ζ_2. δ is the number of divisions that we performed in the path ζ_1. In this case, we have split the paths into slots of 10 cm in which we calculate the area. The paths are more similar when the dissimilarity approaches zero.

Figure 6.4 shows the dissimilarity values (along with the standard errors in the case of the 5 paths of RRT*-based approximations) for the trials 1, 2 and 3 respectively. As can be seen, for trial 1, the dissimilarity of the RTIRL method is the lowest in most of the cases. The other algorithms lead to big dissimilarity values in some scenarios while the RTIRL method keeps a low dissimilarity except for scenarios 5 and 6 where none of the algorithms can plan a good path close to the expert's ones. In the case of the trial 2, a similar behavior arises again. The MMP weight set is not able to reproduce the behavior in scenarios 2, 8 and 10 correctly. Also, the RLT* weight set fails in scenarios 3 and 8. The RTIRL algorithm only partially fails in scenario 8 because some of the 5 paths take another homotopy. Finally, for trial 3, the RTIRL was able to reproduce very well all the evaluation paths while the MMP and RLT* cost functions lead to paths farther from the ground-truth in scenarios 1, 4 and 8, and 4 and 6 respectively.

For a better understanding of the dissimilarity values, Figure 6.5 shows a visual comparison of the paths corresponding with some of the worst cases of the scenarios for evaluation observed in the dissimilarity comparison presented in the Figure 6.4. As can be seen, the weight set learned by the RTIRL approximation (blue lines) leads the paths closer to the demonstrated ones (green circles) in most of the cases.

Table 6.3 summarizes the dissimilarity values for the 3 trials. The results obtained by the weights learned with the RTIRL algorithm are better than the state-of-the-art methods.

Table 6.3: Dissimilarity means and standard errors committed by the learning algorithms in the three trials

Dissimilarity	RTIRL	RLT*	MMP
Trial 1	$\mathbf{0.027 \pm 0.014}$	0.065 ± 0.024	0.098 ± 0.028
Trial 2	$\mathbf{0.010 \pm 0.004}$	0.025 ± 0.011	0.037 ± 0.022
Trial 3	$\mathbf{0.007 \pm 0.000}$	0.024 ± 0.011	0.061 ± 0.027
Mean:	$\mathbf{0.015 \pm 0.006}$	0.038 ± 0.015	0.049 ± 0.025

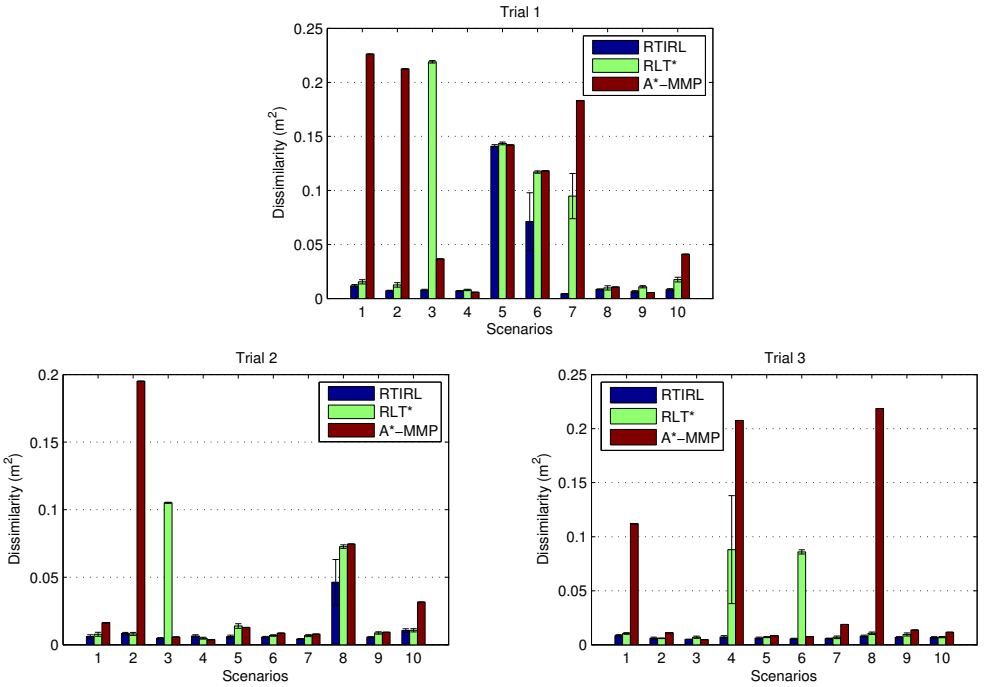


Figure 6.4: Dissimilarity values for the 10 configurations for evaluation of the 3 trials.

6.4.4 Feature count comparison

Another interesting comparison is the feature count of the paths $f(\zeta)$, obtained according to equation (6.2). Figure 6.6 shows the relative errors committed in the feature counts of the paths concerning those of the demonstration paths. Again, the RTIRL method commits the lowest errors in most of the cases while the other methods fail strongly in some cases. These results are stated in Table 6.4, which averages the results of the 3 trials. As can be seen, the RTIRL method reaches significant better results than the RLT* and MMP methods which performance is quite similar according to this metric.

6.4.5 Path cost comparison

Finally, as the weights of the ground-truth cost function are known, we can also compare the costs of the learned paths and the demonstration paths. To do that, we can calculate the relative error in the cost of the paths according to:

$$RE_{costs}(\zeta_l^i, \zeta_d^i) = \frac{\omega_{gt}^T(f(\zeta_l^i) - f(\zeta_d^i))}{\omega_{gt}^T f(\zeta_d^i)} \tag{6.22}$$

Figure 6.5: Visual comparison of the paths learned in some of the scenarios for evaluation of the three trials.

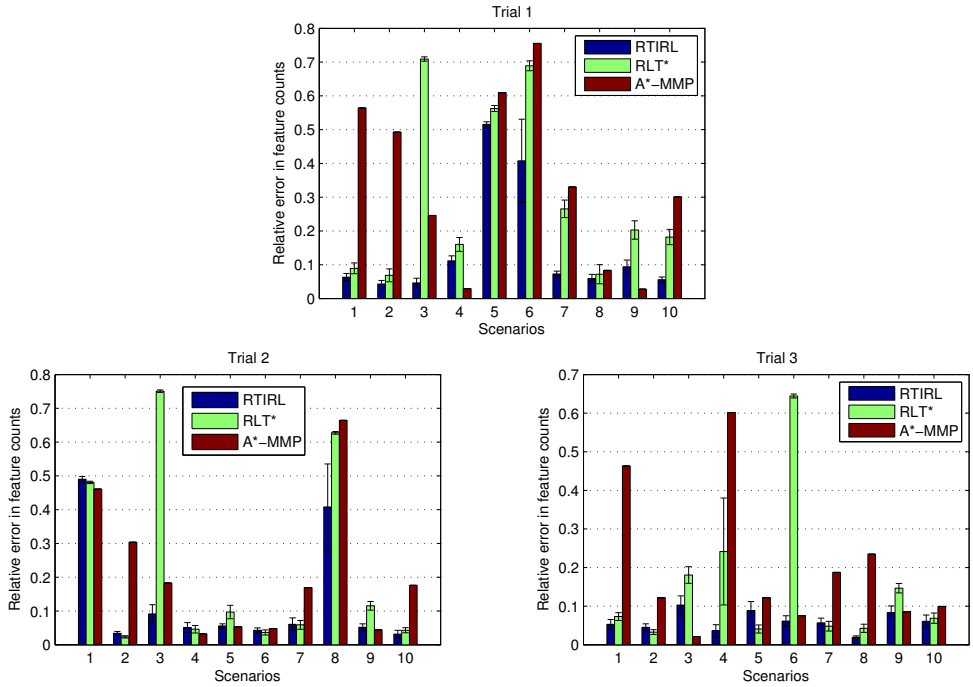


Figure 6.6: Relative error in the feature counts for the 10 configurations for evaluation of the 3 trials.

Table 6.4: Mean relative errors in the feature counts of the paths with the standard errors committed by the learning algorithms in the three trials

RE_{fc}	RTIRL	RLT*	MMP
Trial 1	$\mathbf{0.147 \pm 0.054}$	0.300 ± 0.080	0.344 ± 0.081
Trial 2	$\mathbf{0.132 \pm 0.054}$	0.228 ± 0.088	0.214 ± 0.066
Trial 3	$\mathbf{0.061 \pm 0.008}$	0.152 ± 0.059	0.201 ± 0.059
Mean:	$\mathbf{0.113 \pm 0.038}$	0.227 ± 0.076	0.253 ± 0.069

where ζ_l^i is the path obtained in the scenario i with the cost function learned and ζ_d^i is the demonstration path in the scenario i. ω_{gt} is the weight vector of the ground-truth function.

The relative errors in the costs of the paths for the evaluation scenarios of the 3 trials are shown in Fig. 6.7. As can be seen, the results are quite similar to those obtained for the relative error in the feature count, where the error committed by the RTIRL algorithm is much lower than the error reached by the other approximations. Moreover, the average results of the trials are presented

in Table 6.5. Again, the RTIRL improves the results of the RLT* and MMP methods.

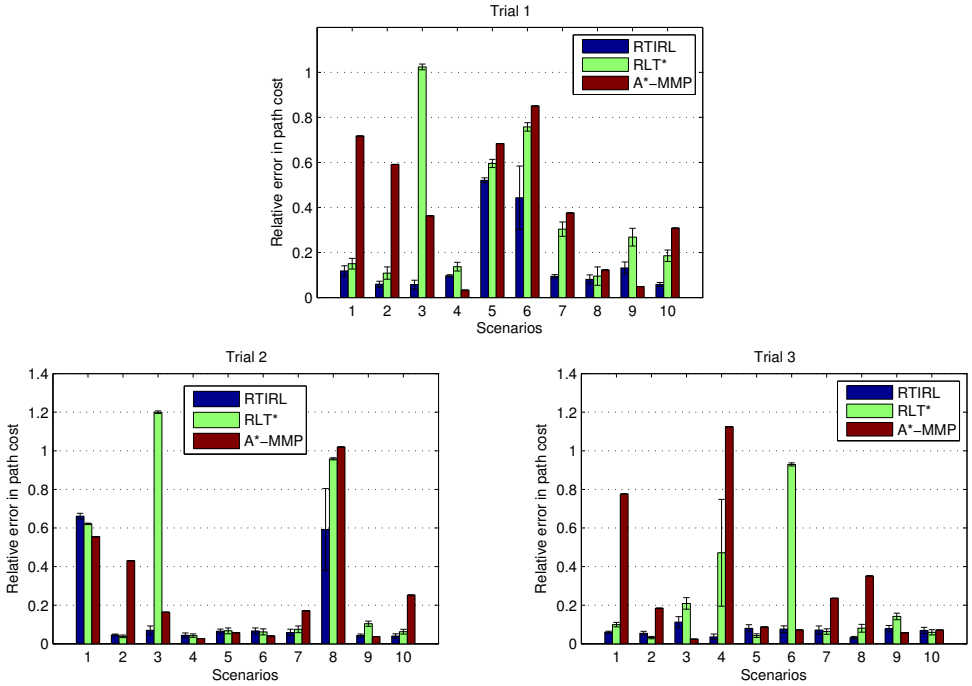


Figure 6.7: Relative error in the cost of the paths for the 10 configurations for evaluation of the 3 trials.

Table 6.5: Mean relative errors in the path costs with the standard errors committed by the learning algorithms in the three trials

RE_{costs}	RTIRL	RLT*	MMP
Trial 1	$\mathbf{0.166 \pm 0.054}$	0.363 ± 0.101	0.410 ± 0.092
Trial 2	$\mathbf{0.169 \pm 0.077}$	0.323 ± 0.139	0.276 ± 0.100
Trial 3	$\mathbf{0.067 \pm 0.007}$	0.213 ± 0.090	0.299 ± 0.116
Mean:	$\mathbf{0.134 \pm 0.046}$	0.300 ± 0.110	0.328 ± 0.103

6.4.6 Statistical significance

A Welch's T-test is employed to compare the results of the methods in the 3 trials. Table 6.6 shows the results of the t-tests when comparing the different metrics

along with the p-values obtained. As can be observed, the better results reached by the RTIRL method can be considered significant (for a 0.05 level) whereas the differences between the RLT* and the MMP method can be neglected in the setup presented.

Table 6.6: Results of the Welch's t-test

T-tests	RTIRL-RLT*	RTIRL-MMP	RLT*-MMP
Dissim.	$\mathbf{p = 0.041}$	$\mathbf{p = 0.003}$	$p = 0.124$
RE_{fc}	$\mathbf{p = 0.031}$	$\mathbf{p = 0.005}$	$p = 0.663$
RE_{costs}	$\mathbf{p = 0.024}$	$\mathbf{p = 0.006}$	$p = 0.828$

6.4.7 Learning with different RRT* planning times

We also evaluate the performance of the proposed learning algorithm regarding the time that the RRT* planner is allowed to plan a path during the learning process (this time may be different from that used in execution after learning) and the number of repetitions of the planning. Table 6.7 shows the average relative error committed in the weights for the methods RTIRL y RLT* according to different planning times of the RRT* planner (0.5 *seconds*, 3 *seconds* and 9 *seconds*) and different number of planning repetitions (10 repetitions, 5 repetitions and 1 repetition respectively). As can be observed, the algorithm behaves well even for short planning times. The relative error in the weights just varies around a 3 % in both algorithms. These results highlight the importance of the number of repetitions. Even if the time for planning is short, we can still have a good estimation of the average features values if we repeat the plan several times. The results also show that the RTIRL is all in all more efficient in terms error vs. planning time than the RLT* algorithm.

Table 6.7: Average relative error in the learned weights committed by the learning algorithms according to different planning times of the RRT* planner used in the learning phase

RE_{ω}	0.5 s (10 rep)	3 s (5 rep)	9 s (1 rep)
RTIRL	**0.098**	**0.068**	**0.077**
RLT*	0.149	0.171	0.180

6.5 Conclusions

In this chapter, a novel approach for teaching a robot social navigation behaviors using demonstrations was presented. To this end, a method based on IRL has been implemented and linked with a regular RRT* to learn the weights of its cost function, so the planner behaves similarly to the demonstrated behaviors. The method is simple to implement and allows to overcome the typical problems associated with IRLs based on MDPs. The proposed method allows to deal with continuous state spaces and is capable of dealing with larger scenarios (regarding persons and features) than previous discrete-MDP-based approaches for human-aware navigation (Ramón-Vigo et al., 2014; Pérez-Higueras et al., 2014), as seen in Chapter 3. It also simplifies the generalization of the behavior thanks to the inherent benefits of RRTs. The approach has been validated using demonstrations provided from a ground-truth cost function. The results showed that it can properly approximate the demonstrated cost function, and to obtain behaviors more similar to the demonstrations than related algorithms from the state-of-art. Moreover, this approach has been successfully applied in the real mobile robot of the European project TERESA, as is presented in the next Chapter.

However, this approach still requires the representation of the cost function as a weighted linear combination of features, which can limit the learning of complex behaviors that can not be properly described with a linear combination. Furthermore, we have to manually identify the set of relevant features involved in the task, which is not easy to determine in problems such as human-aware navigation. These drawbacks are addressed in Chapter 8, in which some solutions are proposed.

Chapter 7

TERESA: a social navigation system in telepresence robots for elderly

"It's alive! It's alive!"
Frankenstein, 1931

In this Chapter, the integration of the research work presented in Chapters 5 and 6 into the navigation system of the TERESA[1] European project is presented. The extensive evaluation and results of different experiments performed with the real telepresence robot are also showed.

TERESA aims to develop a telepresence robot of unprecedented social intelligence, thereby helping to pave the way for the deployment of robots in settings such as homes, schools, and hospitals that require substantial human interaction. In telepresence systems, a human controller remotely interacts with people by guiding a remotely located robot, allowing the controller to be more physically present than with standard teleconferencing. The TERESA project has developed a new telepresence system that frees the controller from low-level decisions regarding navigation and body pose in social settings. Instead, TERESA has the social intelligence to perform these functions automatically (Shiarlis et al., 2015). The ultimate goal of the project is to deploy this system in an elderly day centre to allow elderly people to participate in social events even when they are unable to travel to the centre.

7.1 Introduction

Telepresence robots are gaining an important role in robotics (Kristoffersson et al., 2013). These robots are called "Skype on a stick" because they combine the conversation capabilities of teleconference software and the mobility of the robots

[1]TERESA: TElepresence REinforcement-learning Social Agent. Project reference: FP7-ICT-2013-10. Web: `http://www.teresaproject.eu`

Figure 7.1: Image of the telepresence robot developed in the TERESA project navigating among people.

controlled by humans to allow for better social interaction. In particular, we consider the application of telepresence robots for elderly (Shiarlis et al., 2015). However, maintaining a social interaction and, at the same time, controlling the low-level navigation of the robot overloads the cognition level required by the user, which is especially critical for elderly. That may lead to mistakes in navigation and to pay less attention to the interaction and communication tasks (Tsui et al., 2011). Actually, partial autonomy regarding navigation is a feature requested by telepresence users according to studies like in (Desai et al., 2011) and the findings of the user studies for the requirements of TERESA (TERESA Consortium, 2014).

Therefore, the main goal of the TERESA project is to develop a semi-autonomous telepresence robot for elderly centers, increasing its autonomy in such a manner that users are relieved from lower-level navigation tasks. At the same time, the social intelligence of the robot is improved, providing additional functionalities of interest for the application, such as navigating autonomously to pre-defined goal points, socially approaching people, or walking side-by-side during a conversation. Figure 7.1 shows a capture of the telepresence robot navigating autonomously in a crowded scene.

Figure 7.2: Final disposition of sensors on TERESA. The sensors used for navigation are the LIDARs (there is another one in the back, not shown here) and the XTion cameras.

This Chapter firstly presents the architecture for the social navigation system of TERESA, which is a two-level decision-making system which combines different social behaviors (encoded as macro-actions) intelligently. Learning algorithms (such as the techniques presented in Chapters 5 and 6) are integrated at the macro-action level to endow the robot with social navigation skills and different behaviors. Secondly, the thorough evaluations and experiments performed with the robot in real environments are presented.

7.2 TERESA architecture for social navigation

The TERESA architecture devised for human-aware navigation is presented in this section. So, key features like the sensors employed, the system of people localization and the navigation architecture are introduced here.

7.2.1 Hardware of TERESA robot

The TERESA robot is an evolution of the original Giraff commercial telepresence system[2], in which a new microcontroller and a new shell (produced by the com-

[2] http://www.giraff.org/?lang=en

pany IDMind[3]) have been designed. Furthermore, the robot has been equipped with several sensors that provide appropriate environment measurements to the software stack. Thus, some of them are used for safe navigation (i.e., obstacle avoidance), whereas others are used for retrieving information about the social scene around.

Figure 7.2 shows the location the sensors on the robot frame. We have defined two main groups to cluster them by their aim, namely navigation and interaction sensors:

LIDARs [4] Two scanning laser rangefinders for obstacle detection and localization of the robot. They are located at the front and the back of the robot and cover 360 around it. They are also used to obtain person detections.

XTion cameras [5] These cameras use infrared sensors, adaptive depth detection technology and color image sensing. One is tilted 40 degrees approximately, thus providing the navigation stack with readings that can be useful for 3D obstacle detection, such as steps, holes or table's boards. The second one, facing forward, is also used for person detection. It helps to differentiate simple obstacles as furniture or walls from people. This camera is used for navigation and interaction purposes.

Microphone [6] This sensor is used as a voice tracker and provides an omnidirectional field of view for detecting the active talker. It is used for interaction purposes.

HD camera [7] It provides a high-resolution video streaming which is analyzed in real time to detect the emotions that can arise while the robot performs its tasks. It is used for interaction purposes.

The other devices depicted in Fig. 7.2 are aimed at enhancing the user experience, so they provide the users with video streaming (existing camera) and audio (speakers). Users can also interact with the robot by selecting the volume or through the yes/no buttons.

Lastly, the charging of the robot can be done by plugging the manual charger or using a charging station that makes contact with the charger pads depicted in Fig. 7.2.

[3] http://www.idmind.pt/

[4] https://www.hokuyo-aut.jp/02sensor/07scanner/ust_10lx_20lx.html

[5] https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/

[6] https://www.acousticmagic.com/products/voice-tracker-ii-details/

[7] http://www.teledynedalsa.com/imaging/products/cameras/area-scan/genie/CR-GM0X-H140X/

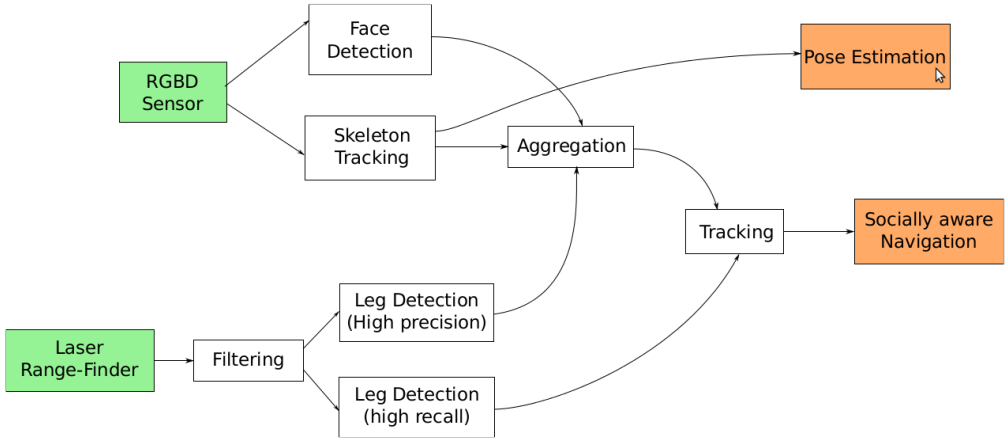


Figure 7.3: Schematic representation of the person localization and tracking modules of the TERESA project.

7.2.2 People localization and tracking

One of the modules that is required for human-aware navigation and interaction with people is the capacity to detect and localize them. We briefly introduce here the localization component for multiple people detection and tracking employed in the TERESA project. This work was mainly led by the Human Media Interaction Group of the University of Twente, as a partner of the project. The Pablo de Olavide University collaborated in the data fusion and people detection with laser range finders, as we explain down below. Further details can be found in the Deliverable 2.2: Pose Estimation, of the project (TERESA Consortium, 2016a).

Person localization must be achieved all around the robot under challenging circumstances of occlusion and difficult lighting conditions to exhibit the desired social behavior in the robot. This localization is based on data from the following sensors, commented previously: one colour+depth camera (RGB-D sensor) and two laser rangefinders.

That is achieved by combining different existing algorithms for localization:

- **Skeleton tracking implemented for RGB-D sensors**. The ROS COB OpenNI-based skeleton-tracking module[8] is employed for this. The centre of mass of the skeleton is used as the position, which is integrated with the other detections for tracking. The advantages of the skeleton tracking are that it relies on many detectors for body parts, and therefore has a

[8] `http://wiki.ros.org/cob_openni2_tracker`

"holistic" view of the person. However, the skeleton tracker does have some weaknesses. Most importantly, when the robot is moving, its detections are not very reliable, and it is easily confused.

- **Face detection algorithms from RGB-D sensors**. To deal with the weaknesses of the skeleton tracker, the detection of heads and faces were also considered through the ROS COB people detection module[9]. Using the active depth sensor allows the system to perform well even in poor lighting conditions, and allows for high-precision, high-recall detection of people when only their head is visible in front of the robot. Moreover, the face detections are not measurably affected by the robot motion.

- **Leg detectors from the laser-range finders**. The combination of skeleton and face detectors result in very high-quality person detection in the field of view of the RGB-D sensor, which is the area in front of the robot. Nevertheless, to implement acceptable social behavior, our modules do require tracking the position of all the people surrounding the robot. To achieve person detection in the complete 360° around the robot, the ROS module based on the people's legs detector presented in Arras et al. (2007) is employed. Two instances of the module are set up, where the parameters of one detector are set to obtain high recall at the expense of a lower precision while the other detector maintains a higher precision at the expense of a comparatively lower recall. The false detections are handled in the aggregation and tracking modules.

An overall overview of the architecture is shown in Fig. 7.3. It shows how the RGB-D sensor data is used for face detection and skeleton tracking. These are aggregated with the leg detections and fed to the tracker. The tracker is additionally given the output of a separate leg detector which detects many more people at the expense of more miss-detections; these detections are used to sustain existing tracks but not to initiate new tracks.

The people detections are initiated with a Kalman filter for each person. When new detection data is received from any source, it is matched with the position prediction given by the Kalman filters. Thus, we associate the information from different sources, keeping track of the current people in the scene, and also adding new ones and/or deleting the missing ones.

7.2.3 Navigation architecture

Conceiving a single navigation function able to deal with all the human-aware navigation problems is very challenging. Many existing approaches only address

[9] `http://wiki.ros.org/cob_people_detection`

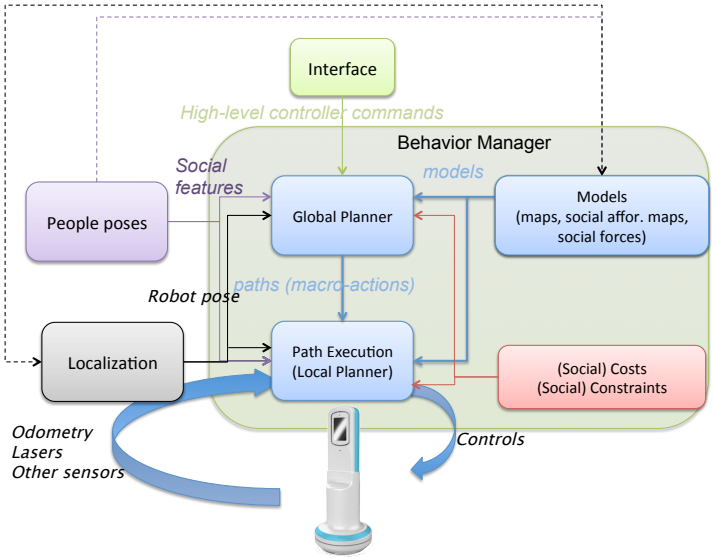


Figure 7.4: A sketch of the general architecture for the navigation macro-actions. The TERESA architecture adds new models, social costs and constraints. Furthermore, the different modules are tailored to each particular macro-action.

one particular problem, like social people avoidance (Henry et al., 2010), people approach (Truong and Ngo, 2016), following a person walking side-by-side (Ferrer et al., 2016), etc.

We concur that it is very complex to derive general models, costs, planners, etc. for every potential situation, as we commented along this thesis, so our navigation architecture reflects this fact. Thus, the navigation stack offers a set of social **macro-actions** that can be triggered by the robot user or executed automatically when the social situation requires. Each macro-action involves different models and algorithms, like those developed in Chapters 5 and 6.

We have used the ROS middleware[10] to implement the navigation stack that has been developed in the project. This stack substitutes completely the *move_base* ROS navigation stack. It follows the same classical navigation architecture, as we introduced in Chapter 2 and also followed in the architecture of FROG (Chapter 4), but adding the components for human-aware navigation. Figure 7.4 shows a simplified block diagram of the architecture.

The macro-actions mentioned above are implemented following a similar scheme. Besides receiving the information from the sensors and the perception modules, the main components are a global or high-level planner, that provides global paths or actions; and a local planner/controller also incorporating

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

dynamic obstacle avoidance. Each macro-action may involve different social cost/restrictions and/or different models or implementations of the planners. When the Behavior Manager activates a particular macro-action, the navigation stack loads the required modules and models. That fits the objectives of TERESA and allows the social models, cost functions, and planners to be learned from data for that specific application.

The navigation macro-actions considered are the following:

- **Navigate to Waypoint**, in which the robot navigates socially to the desired waypoint, and that is activated with a command from the visitor at the control station by selecting from a waypoint list. The method described in Chapter 6 is employed to this aim.

- **Navigate to Interaction Target**, in which the robot approaches a person to interact with him/her socially. The person is selected by the visitor at the control station by clicking the person on the image. This person is denoted as the Interaction Target. The approximation presented in Chapter 5 is used to achieve this.

- **Walk side-by-side**, where the robot tries to accompany a person walking together to other room or place while in conversation. That is also triggered by the visitor.

- **Assisted steering** This macro-action is activated whenever the visitor tries to drives the robot manually through his/her control interface. This assistant is in charge of evaluating the velocity commands sent by the visitor, checking possible collisions and generating smooth valid commands when possible (stopping in case of unavoidable collision).

The previous macros are activated by the user of the robot (explicitly or implicitly). Moreover, there is another group of macro-actions that are triggered by events independent of the user's actions:

- **Yield**. While the robot is navigating, if any person stands on the robot's path within a narrow space like a doorway, the robot will yield *politely*, retreating to a position that allows the person to come through first. Once the path is free again, the robot will resume its previous action.

- **Wait**. This macro is activated whether the robot finds its path blocked while navigating. To avoid a collision, the robot keeps standing still for a pre-defined time, trying to resume navigation after a waiting time.

- **Navigate Home**. This macro-action is triggered whenever the battery level of the robot is under a certain threshold, or the visitor ends the call. In that case, the robot should go back to a pre-defined position where the docking station is installed to charge (*Home*).

7.3 Architecture implementation

The navigation architecture of TERESA has two levels of decision making. At the highest level, there is a *Behavior Manager* in charge of selecting the different macro-actions and controlling the transitions between them; at the lower level, each macro-action represents a navigation behavior.

Hereafter, the behavior manager is briefly introduced, and the integration of the work presented in Chapters 5 and 6 into the macro-actions architecture is emphasized. The implementation of the different modules of the navigation architecture can be found, under BSD license, in the ROS meta package *upo_robot_navigation* in the Github of the UPO Service Robotics Lab[11].

7.3.1 Behavior manager

The *Behavior Manager* represents the top level decision-making component, and is in charge of controlling navigation functionalities through the concept of macro-actions, which can be implemented in different manners, such as Finite State Machines (FSM), control-based, decision-theoretic-based, etc. The whole architecture is based on the ROS middleware. Each macro-action must provide an interface based on the ROS actionlib[12] for the rest of the system, acting as services. These macro-actions are then activated when certain events happen and/or inputs are received from the user. The Behavior Manager is encoded as a Finite State Machine (FSM) that produces the adequate transitions between macro-actions. In particular, we have used SMACH [13] for the implementation.

Figure 7.5 shows the part of the general FSM diagram corresponding to the eight navigation macro-actions implemented, along with the possible transitions between them according to the events that may occur. The initial macro-action is "Wait for goal", where the robot waits for an event that initiates a transition. The transition to the *interaction in conversation* states is done through the event "Nav succeeded".

[11] `https://github.com/robotics-upo/upo_robot_navigation`
[12] `http://wiki.ros.org/actionlib`
[13] `http://wiki.ros.org/smach`

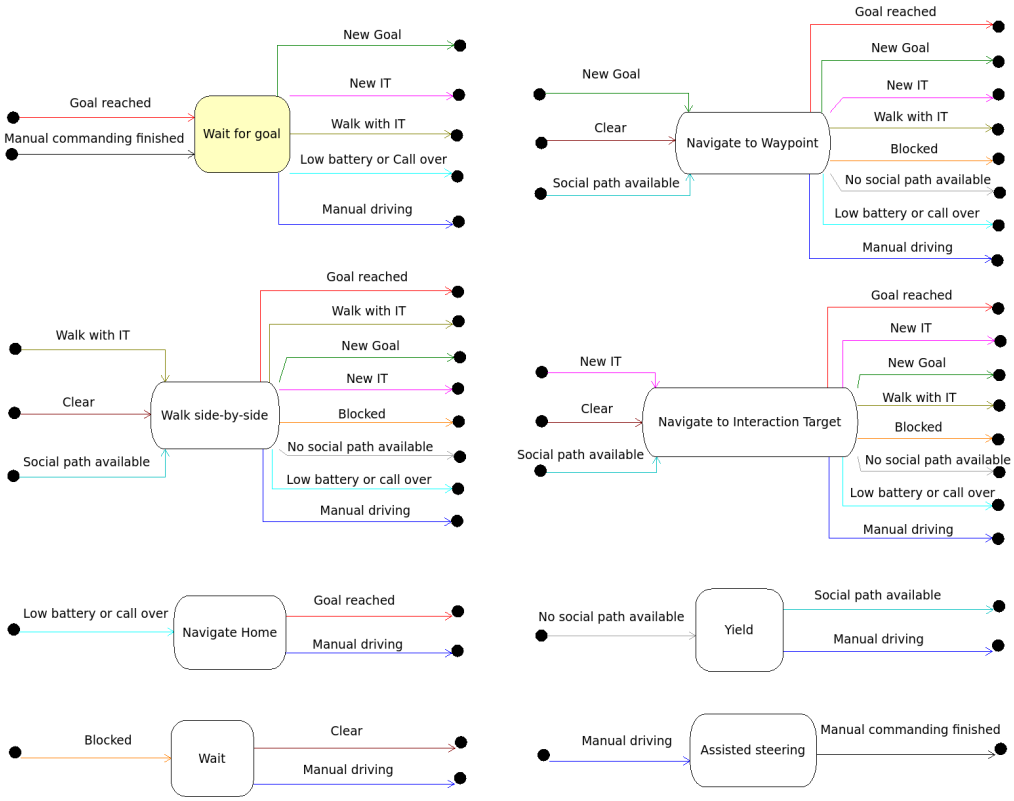


Figure 7.5: Diagram of the finite state machine for social navigation. The initial state is shown in yellow.

7.3.2 Social navigation to waypoint

The first macro-action that was implemented is the navigation between waypoints in the scenario. Thus, the robot has to show a socially-acceptable behavior while encountering people on its way to the goal.

To achieve this, the learning approach presented in Chapter 6 is considered here to endow a local RRT* planner with a social navigation behavior. The set of features designed for human-aware navigation and also presented in Chapter 6 is employed to fulfill the task.

The macro itself is implemented using a classical three-level architecture: a global planner computes a global path, then local planning and a control layer deal with dynamic obstacle avoidance.

On top of it, an A* planner provides a global path towards the destination. The relation between the two levels is the following:

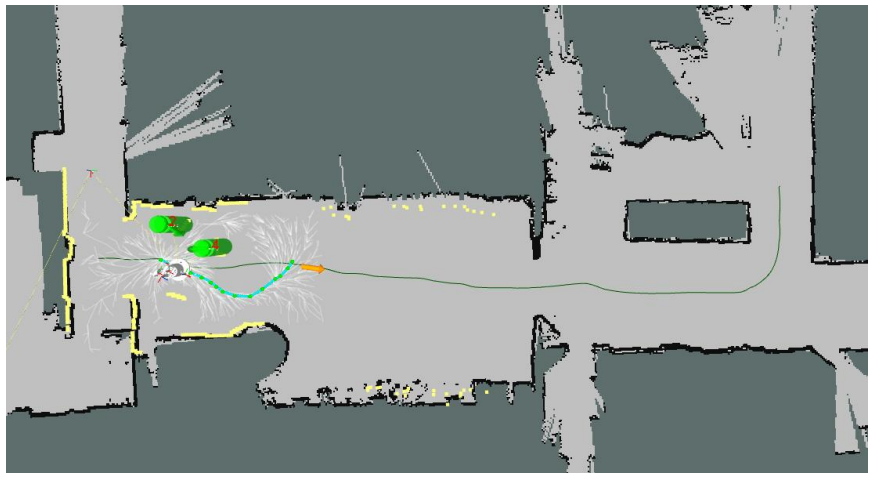


Figure 7.6: Capture of the navigation simulator for describing the planning architecture. The dark green line represents the path obtained by the A* planner to the final goal. The light green line represents the RRT* path in a local area. The light yellow line represents the lasers sensors readings. The orange arrow represents the current local goal of the RRT* planner. Finally, the green cylinders marked 2 and 4, represent people detected in the vicinity of the robot.

- Firstly, the A* global planner is employed to obtain a path from the initial robot position to the final goal. The plan is calculated in the map frame over the static map of the scenario, no-mapped static obstacles and dynamic obstacles are not taken into account. This path is computed only one time when a navigation goal is received.

- Secondly, the RRT* planner uses the previous plan to determine a local goal. This sub-goal is determined as the intersection point of the A* plan and a local area around the robot ($10x10\ m$ approximately with the robot in the center). Then, the RRT* plan a local path on the robot frame towards this local goal taking into account all the people and static and dynamic obstacles in the vicinity of the robot. Moreover, the RRT* space sampling is partially biased to areas close to the A* plan. The sub-goal is updated and the RRT* path is re-planned at a frequency between $1 - 2\ Hz$. Figure 7.6 shows a descriptive example of this two-level planning architecture.

Finally, a low-level controller is employed to send the correct commands to the robot to follow the RRT* path. It based on pure pursuit path tracking to command velocities to the differential robot to follow the path smoothly. Moreover, it has been extended to perform a collision detection checking similar to

the Dynamic Windows Approach algorithm (Fox et al., 1997). If the forward projection of the robot movement given by the control law is detected as a possible collision, a valid command is tried to be found by sampling small variations of the given angular velocity. If a possible collision is still detected, a rotation in place in the correct direction is performed to avoid very close obstacles. Moreover, a linear decrease of the velocities when approaching the goal has also been added.

The use of the RRT* planner in the scheme presented has some advantages. In one hand, it plans in a fixed-size continuous space unlike the A* planner who needs a discretized space and its performance depends on the resolution of the discretization. On the other hand, the random component of the planner allows to learn and plan paths of different homotopies that are similarly valid for avoiding obstacles or people.

7.3.3 Social people approaching

The navigation to an interaction target requires that the robot approaches the person taking into account the orientation of the person and re-calculating the goal if the person moves or changes his position.

To perform the approaching person socially, a set of human trajectories approaching other people has been used to learn the task, as can be seen in Fig. 7.7. In most of the approaching angles, two valid homotopies to reach the goal have been considered. That gives more flexibility to the RRT* planner in case one of the homotopy paths is blocked for any reason. In particular, a model in feature space has been learned from the human demonstrations and encoded as a Gaussian Mixture Model (GMM), as explained in Chapter 5. Then, this GMM is employed to bias the sampling of the Optimal-RRT planner to direct the planner to the correct areas to perform the approaching.

The steps followed when the macro-action of navigation to an interaction target is triggered, are listed here:

1. The distance to the target is checked. If it is less than 3.20 *meters*, the GMM sampling process is activated.

2. If the GMM sampling is active, the angle between the robot position and the heading of the interaction target is discretized in one of the eight categories of possible approaching, as shown in Fig. 7.7.

3. The corresponding GMMs (in two valid homotopies in some cases) are selected according to the previous step and a 90 % of samples used in the RRT* planner are then drawn from the GMMs.

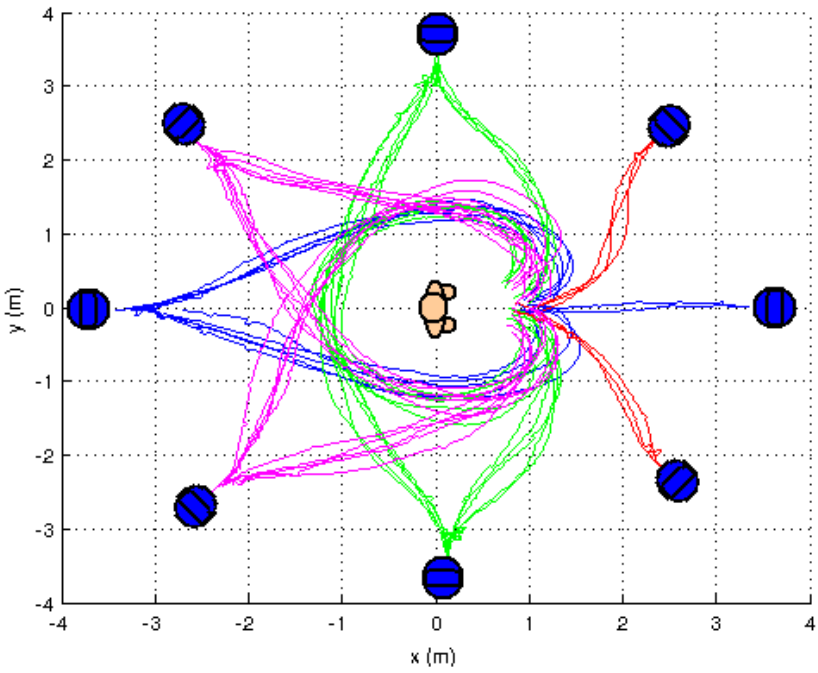


Figure 7.7: Real trajectories employed to learn how to approach a person for the TERESA robot. Trajectories for approaching angles $[0, \pm\pi]$ rad are shown in blue lines. $[\pi/4, -\pi/4]$ rad in red lines. $[\pi/2, -\pi/2]$ rad in green lines. Finally, $[3\pi/4, -3\pi/4]$ rad in magenta lines.

4. At a frequency of 2 Hz approximately, the previous steps are checked again the sampling process is updated if necessary.

7.4 Evaluations

In this section, the waypoint navigation macro-action is evaluated by performing real experiments in the laboratory. First, the navigation functionality will be assessed with an evaluation inspired by the benchmark of the European Robotic League for Social Robots (ERL-SR) used in Rock-In competition[14]. Secondly, an evaluation of the "social" capabilities of the system will be studied. In both cases, a comparison with a standard navigation stack where all entities in the scene are simple obstacles is performed. To do that, the move_base framework

[14] `http://rockinrobotchallenge.eu/`

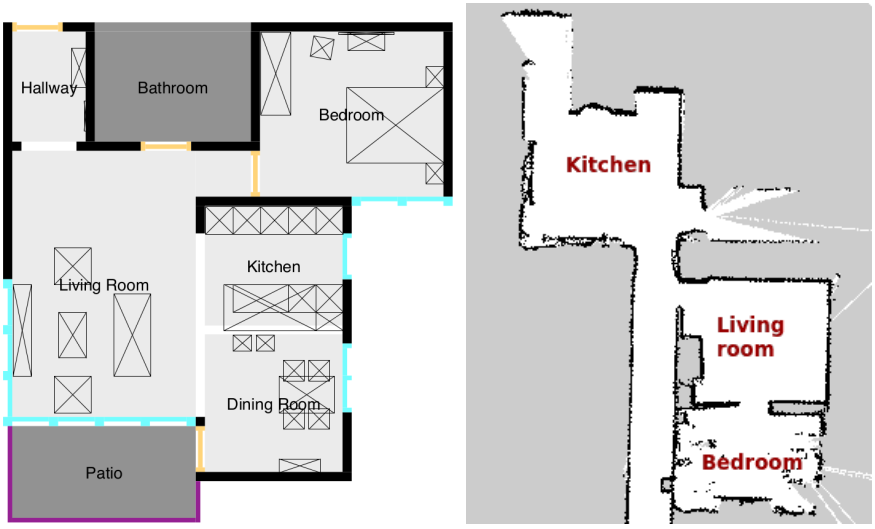


Figure 7.8: Scenarios for navigation evaluation. Left, testbed of the Rockin Home Competition. Right, testbed in the Pablo de Olavide University.

from ROS is employed[15].

7.4.1 Benchmarking according to ERL-SR

According to the ERL-SR Rulebook[16], the navigation functionality benchmark assesses the robot's capability to correctly, safely, and autonomously navigate in an ordinary apartment. The task includes the navigation in an apartment-like environment with furniture, walls, and doors, i.e., in a previously mapped area; avoiding collisions with a different type of unknown obstacles, in unknown positions (not previously mapped); and navigate in the presence of people in the arena. The robot receives a list of waypoints it has to follow in respective order.

The proposed testbed of the Rockin Home Competition is depicted in Figure 7.8. A similar apartment environment has been replicated at the Universidad Pablo de Olavide (Figure 7.8, right). In the living room area, some unmapped obstacles have been placed. Also, the kitchen area counts with the presence of people wandering around. A set of waypoints have been defined in the scenario, as indicated in the Figure 7.9. The robot has to reach each waypoint following the order established.

The evaluation of the navigation takes into account the following three met-

[15] http://wiki.ros.org/move_base

[16] http://sparc-robotics.eu/wp-content/uploads/2016/04/ERL_Service.compressed.pdf

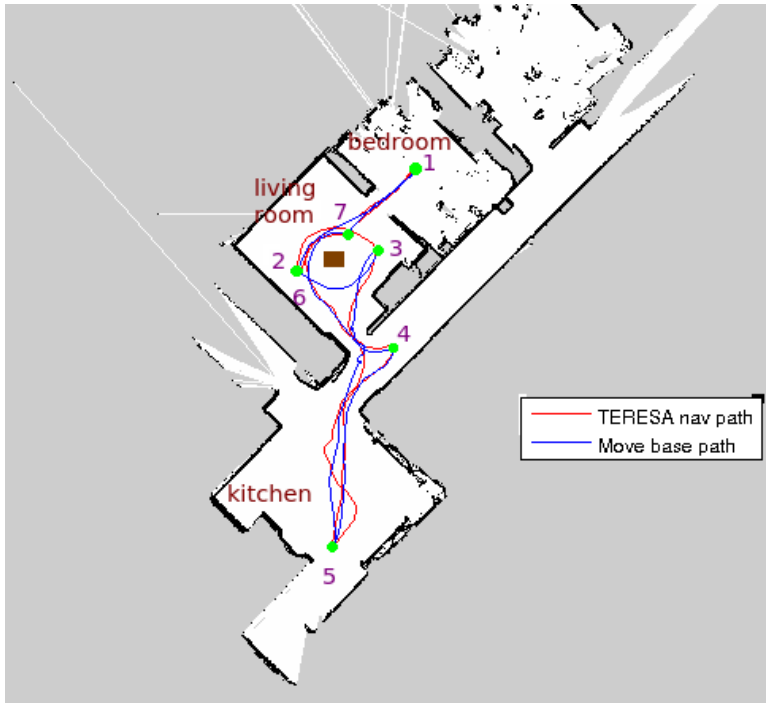


Figure 7.9: Waypoints (green dots) employed for navigation functionality evaluation. An example of the paths followed by the TERESA system (red line) and the move_base system (blue line) are also showed.

rics:

- The distances between the robot's position and the respective position of the waypoint (both the Euclidean distance between the waypoint and the robot in meters (P_{xy}) and the difference in the orientation in radians (P_{yaw})).
- The time spent by the robot to go from each waypoint to the next waypoint in seconds (T_p).
- The number of times that the robot hits each obstacle ($Hits$).

We also have added an extra metric that is the total path length in meters (L_p). Three runs are carried out with each one of the systems. Table 7.1 shows the metrics (in average) with their standard deviations of the three runs for each of the paths between waypoints. As can be seen, there are not very significant differences between the two navigation systems. As expected the time to reach the goal for the most of trajectories is a bit shorter in the case of move_base since

it tries to minimize the distance and time to reach the goal, unlike the TERESA system which also takes into account the people comfort. Anyway, the differences are small. Moreover, the precision on reaching the waypoints is slightly better in the case of move_base by a small margin. Both navigation systems successfully reached all the goal without any hits.

Table 7.1: Rockin evaluation metrics for each waypoint

TERESA	T_p	P_{xy}	P_{yaw}	L_p	$Hits$
W1-W2	$\mathbf{17.73 \pm 1.70}$	0.15 ± 0.03	0.25 ± 0.15	5.10 ± 0.11	0
W2-W3	14.95 ± 8.03	0.39 ± 0.25	$\mathbf{0.16 \pm 0.14}$	$\mathbf{4.65 \pm 2.80}$	0
W3-W4	28.89 ± 12.3	0.13 ± 0.02	0.29 ± 0.11	5.58 ± 3.33	0
W4-W5	31.38 ± 18.5	$\mathbf{0.16 \pm 0.02}$	0.36 ± 0.03	6.32 ± 2.88	0
W5-W6	19.07 ± 14.7	0.27 ± 0.22	$\mathbf{0.16 \pm 0.07}$	$\mathbf{4.46 \pm 0.68}$	0
W6-W7	$\mathbf{23.57 \pm 6.10}$	$\mathbf{0.22 \pm 0.08}$	0.29 ± 0.09	$\mathbf{4.89 \pm 3.20}$	0
W7-W1	$\mathbf{15.61 \pm 4.31}$	0.25 ± 0.03	$\mathbf{0.17 \pm 0.14}$	$\mathbf{4.17 \pm 0.83}$	0
MOVEB					
W1-W2	24.47 ± 13.1	$\mathbf{0.10 \pm 0.03}$	$\mathbf{0.09 \pm 0.09}$	$\mathbf{3.17 \pm 0.24}$	0
W2-W3	$\mathbf{14.25 \pm 1.49}$	$\mathbf{0.09 \pm 0.00}$	0.32 ± 0.09	5.00 ± 0.05	0
W3-W4	$\mathbf{18.43 \pm 13.8}$	$\mathbf{0.10 \pm 0.00}$	$\mathbf{0.22 \pm 0.01}$	$\mathbf{3.95 \pm 0.85}$	0
W4-W5	$\mathbf{17.82 \pm 4.89}$	0.37 ± 0.43	$\mathbf{0.19 \pm 0.18}$	$\mathbf{5.56 \pm 3.24}$	0
W5-W6	$\mathbf{18.03 \pm 9.98}$	$\mathbf{0.10 \pm 0.01}$	0.30 ± 0.14	5.17 ± 2.62	0
W6-W7	73.48 ± 25.8	0.35 ± 0.02	$\mathbf{0.20 \pm 0.28}$	6.82 ± 2.51	0
W7-W1	25.96 ± 19.0	$\mathbf{0.07 \pm 0.06}$	0.30 ± 0.03	5.12 ± 2.86	0

Table 7.2 summarizes the results of three runs for all the paths performed with each system (TERESA navigation system and ROS Move base system). The precision reaching the waypoints in the distance metric is slightly better in the case of the move_base. However, the TERESA system has a better total average time to finish the trial. We observed during the trials that the move_base system spent more time rotating and trying to reach the correct goal position than TERESA. We also noted that move_base has some problem when the robot is surrounded by people or unknown obstacles because the A^* planner is not able to find a path to the goal. The TERESA's RRT^* local planner is more efficient for local obstacle avoidance. Regarding navigation functionality, we can say that the performance of both systems is quite similar.

7.4.2 Social Evaluation of the navigation system

In this section, the cost function for social navigation learned using the RTIRL algorithm is evaluated by performing real experiments with the TERESA robot

Table 7.2: Rockin evaluation metrics in average

	T_p	P_{xy}	P_{yaw}	L_p	$Hits$
TERESA	**151.20 ± 9.38**	0.22 ± 0.14	0.24 ± 0.12	35.17 ± 0.17	0
MOVEBASE	192.44 ± 20.78	**0.17 ± 0.18**	**0.23 ± 0.13**	**34.79 ± 0.17**	0

in the laboratory. We perform a comparison between the social capabilities of a navigation system employing the cost function learned from a set of demonstrations (which we will denote as TERESA or SNS (Social Navigation System)) and a standard navigation stack where all entities in the scene are considered as simple obstacles. For the latter we employ the popular move_base framework from ROS[17] (Quigley et al., 2009) (denoted as MOVEBASE or Non-SNS (Non-Social Navigation System)). The default configuration parameters have been used but the related ones to the robot footprint, velocities, and accelerations that have been adjusted to the TERESA robot.

The library of RRT algorithms and the rest of necessary modules for the SNS that have been developed by the Ph.D. candidate are available in the Github of the Service Robotics Lab[18] under BSD license. A RRT* replanning each 0.5 *seconds* in a local area surrounding the robot and using the learned costs is employed for motion planning, and a simple controller derived from the Dynamic Windows Approach algorithm (Fox et al., 1997) is used to follow the RRT path.

7.4.2.1 Metrics

A set of simple metrics for a social evaluation used in the literature (Okal and Arras, 2016b), some of them based on the Proxemics theory (Hall, 1966), has been considered here:

- **Time to reach the goal** (T_p). Time since the robot start the navigation until the goal is correctly reached.

$$T_p = (T_{goal} - T_{ini}) \tag{7.1}$$

- **Path length** (L_p). The length of the path followed by the robot from the initial point to the goal position.

$$L_p = \sum_{i=1}^{N-1} \|x_r^i - x_r^{i+1}\|_2 \tag{7.2}$$

[17] `http://wiki.ros.org/move_base`

[18] `https://github.com/robotics-upo/upo_robot_navigation`

- **Cumulative heading changes** (CHC). It counts the cumulative heading changes of in the robot trajectory measured by angles between successive waypoints. It gives a simple way to check on smoothness of path and energy (Okal and Arras, 2016b) so a low value is desirable.

$$CHC = \sum_{i=1}^{N-1} (h_r^i - h_r^{i+1}) \tag{7.3}$$

  Where h_r^i indicates the heading of the robot in the position i. The angles and their difference are normalized between $-\pi$ and π.

- **Average distance to closest person** (CP_{avg}). A measure of the mean distance from the robot to the closest person along the trajectory.

$$CP_{avg} = \frac{1}{N} \sum_{i=1}^{N} (\|x_r^i - x_{cp}^i\|_2) \tag{7.4}$$

  Where x_{cp}^i indicates the position of the closest person to the robot at step i.

- **Minimum and maximum distance to people** (CP_{min} and CP_{max} respectively). The values of the minimum and the maximum distances from the robot to the people along the trajectory. It can give an idea of the dimension of the robot trajectory with respect to the people in the space.

$$CP_{min} = \min\{\|x_r^i - x_{cp}^i\|_2 \mid \forall i \in N\} \tag{7.5}$$

$$CP_{max} = \max\{\|x_r^i - x_{cp}^i\|_2 \mid \forall i \in N\} \tag{7.6}$$

- **Personal space intrusions** (CP_{prox}). This metric is based on the Proxemics theory which define personal spaces around people for interaction (Hall, 1966). These areas are defined as:

  - Intimate. Distance shorter than 0.45 m.
  - Personal. Distance between 0.45 m and 1.2 m.
  - Social. Distance between 1.2 m and 3.6 m.
  - Public. Distance longer than 3.6 m.

  Thus, the metric classifies the distance between the robot and the closest person at each time step in one of the Proxemics spaces (in our case we use

only three: Intimate, Personal and Social+Public), and obtain a percentage of the time spent in each space for the whole trajectory:

$$CP^k_{prox} = \left(\frac{1}{N} \sum_{j=1}^{N} \mathcal{F}(\|x^j_r - x^j_{cp}\|_2 < \delta^k) \right) * 100 \tag{7.7}$$

where N is the total number of time steps in the trajectory, δ defines the distance range for classification defined by $k = \{Intimate,\ Personal,\ Social+Public\}$, and $\mathcal{F}(\cdot)$ is the indicator function. The desirable behavior should lead the robot to spend most of the time in the Social or Public distance range, but this depends on the dimensions of the space and the density of people in the area. So, for a small area shared with people, intrusions in the Personal area are acceptable whereas for big open space the robot should stay in the Social and Public distances. In any case, the intrusions in the Intimate space should be avoided.

- **Interaction space intrusions** (IS_{prox}). This metric is inspired by the work of Okal and Arras (Okal and Arras, 2016b) in formalizing social normative robot behavior, and it is related to groups of interacting persons. It measures the percentage of time spent by the robot in the three Proxemics spaces considered with respect to an interaction area formed by a group of people that are interacting with each other. The detection of the interaction area of the group is based on the detection of F-formations. A F-formation arises whenever two or more people sustain a spatial and orientational relationship in which the space between them is one to which they have equal, direct, and exclusive access (Cristani et al., 2011; Setti et al., 2015).

$$IS^k_{prox} = \left(\frac{1}{N} \sum_{j=1}^{N} \mathcal{F}(\|x^j_r - x^j_f\|_2 < \delta^k) \right) * 100 \tag{7.8}$$

where x^j_f determines the center of the closest formation or group of people f to the robot at step j.

7.4.2.2 Experiments

The social navigation behavior learned using the learning algorithm RTIRL (presented in Chapter 6) is evaluated here. A cost function, using the features described in Section 6.3, has been learned from navigation demonstrations in static

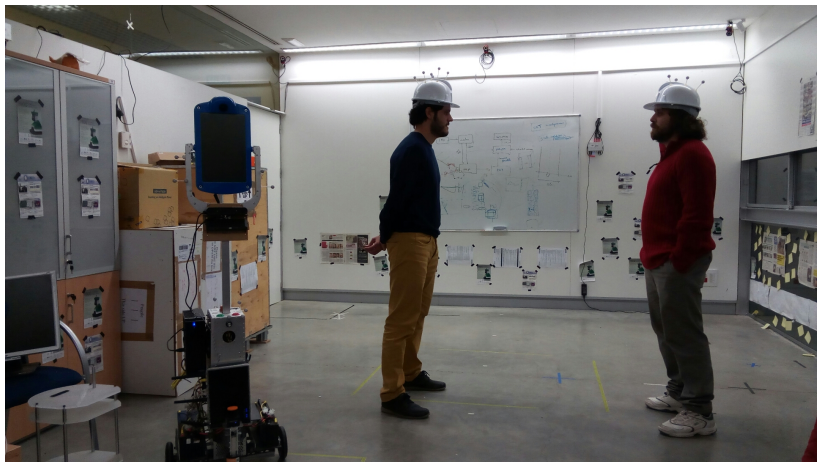


Figure 7.10: Capture of the real experiments for the social navigation evaluation in the University Pablo de Olavide.

scenarios. The demonstrations were recorded in the Robotics Lab and some scenarios of the TERESA project (an elderly care center in Troyes, France, see Fig. 7.16). The planning time for the RRT* employed during the learning phase was 3 *seconds*. The learned cost function is then used by the Social Navigation System.

Two sets of experiments have been performed. One in a static scenario (the learning is performed from static scenes) and another one in a dynamic scenario, to check the behavior in scenarios with moving pedestrians.

These evaluation experiments have been recorded in the Robotics Lab of the Pablo de Olavide University. The room has an approximate dimension of 4.60×7.60 m, and it is equipped with an Optitrack Motion Capture System. This system has been used to accurately estimate the position of the robot and persons involved in the experiments so that results cannot be biased by localization errors.

- **Experiments in static scenarios**
  In the static scenario we have considered two different situations: first, a small group of two people talking to each other in the middle of the scene, as can be seen in Figure 7.10; second, there are two persons in the scenario, but they are not interacting each other. Figure 7.11 shows two examples of the scenarios proposed, where the robot paths and the people position and orientation are shown. As can be seen, the TERESA Navigation System (SNS) avoid interfering with the group indicating a more acceptable behavior than the MOVEBASE Navigation System (Non-

SNS) from the social point of view.

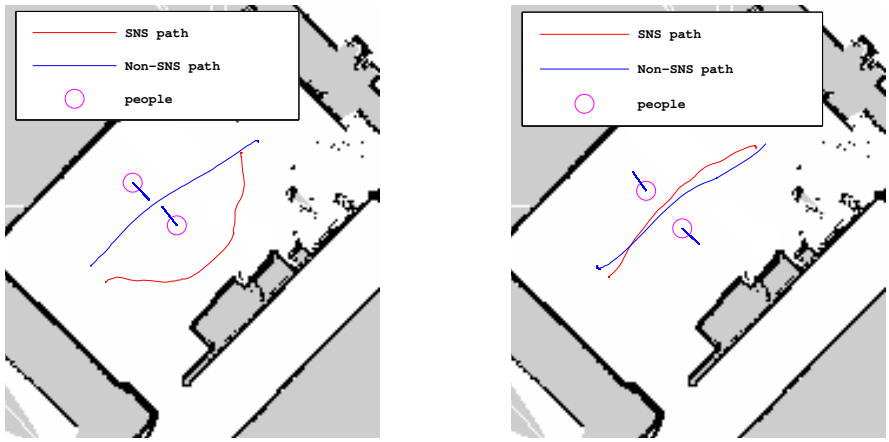


Figure 7.11: Static scenario for social evaluation along with some of the paths obtained by the SNS and non-SNS. Left: Two persons, represented by magenta circles with the blue arrow indicating their orientation, are forming a group. Right: Two persons are facing the walls (not a group). The red line shows the path followed by the TERESA social navigation system (SNS). The blue line shows the path of the MOVEBASE non-social system (Non-SNS).

Tables 7.3 and 7.4 shows the metrics, with their standard deviations, obtained for four runs of each navigation system in the static scenario with a group of interacting people, and another four runs in the scenario with two non-interacting individuals respectively.

In the case of two people not forming a group, the metrics do not present significant differences as can be observed in Table 7.3 (note that the IS_{prox} metric is not applicable because there is no interaction space). That is because of, in this particular setup, the social taught behaviors lead the robot to perform similar paths of the Non-Social MOVEBASE system. Nevertheless, some metrics can be highlighted. The minimum distance to people is higher in the case of TERESA which indicates that the system always was able to keep some distance to the person. And it was also able to spend less time in the Personal space in favor of the time in the Social space which indicates a better social behavior.

Furthermore, the case of a group of people presents more evident differences. The MOVEBASE performs clear intrusions in the Intimate space concerning the closest person and also to the interaction space of the group.

Again, the minimum distance to people is higher in the case of TERESA. These results show that the Social TERESA system is more respectful of the people spaces and the interaction space of the group from a social point of view.

Table 7.3: Results for social navigation evaluation in a static scenario with two people not forming a group

NO-GROUP		TERESA	MOVEBASE
$T_p(s)$		$\mathbf{22.70 \pm 1.52}$	24.18 ± 1.73
$L_p(m)$		$\mathbf{4.86 \pm 0.40}$	4.92 ± 0.23
$CHC(rad)$		4.54 ± 0.79	$\mathbf{4.37 \pm 0.40}$
$CP_{avg}(m)$		1.74 ± 0.09	$\mathbf{1.90 \pm 0.14}$
$CP_{min}(m)$		$\mathbf{1.08 \pm 0.16}$	0.63 ± 0.07
$CP_{max}(m)$		2.47 ± 0.13	$\mathbf{2.82 \pm 0.01}$
$CP_{prox}(\%)$	Intimate	0.00 ± 0.00	0.00 ± 0.00
	Personal	$\mathbf{12.77 \pm 1.02}$	17.24 ± 0.98
	Social+	$\mathbf{87.23 \pm 0.87}$	82.76 ± 0.98

Table 7.4: Results for social navigation evaluation with a static group of two people

GROUP		TERESA	MOVEBASE
$T_p(s)$		23.65 ± 0.64	$\mathbf{23.15 \pm 1.34}$
$L_p(m)$		5.45 ± 0.27	$\mathbf{5.02 \pm 0.08}$
$CHC(rad)$		$\mathbf{4.18 \pm 2.55}$	4.41 ± 0.08
$CP_{avg}(m)$		1.87 ± 0.08	$\mathbf{1.88 \pm 0.12}$
$CP_{min}(m)$		$\mathbf{0.79 \pm 0.17}$	0.42 ± 0.15
$CP_{max}(m)$		2.71 ± 0.11	$\mathbf{2.74 \pm 0.17}$
$CP_{prox}(\%)$	Intimate	$\mathbf{0.00 \pm 0.00}$	2.60 ± 3.67
	Personal	17.83 ± 1.52	$\mathbf{14.50 \pm 3.06}$
	Social+	82.17 ± 1.52	$\mathbf{82.90 \pm 0.61}$
$IS_{prox}(\%)$	Intimate	$\mathbf{0.00 \pm 0.00}$	7.64 ± 0.70
	Personal	$\mathbf{0.00 \pm 0.00}$	13.00 ± 0.01
	Social+	$\mathbf{100.0 \pm 0.00}$	79.37 ± 0.71

- **Experiments in dynamic scenarios**

Another test has been recorded in a dynamic situation. The aim is to check whether the social cost function learned for static situations is still

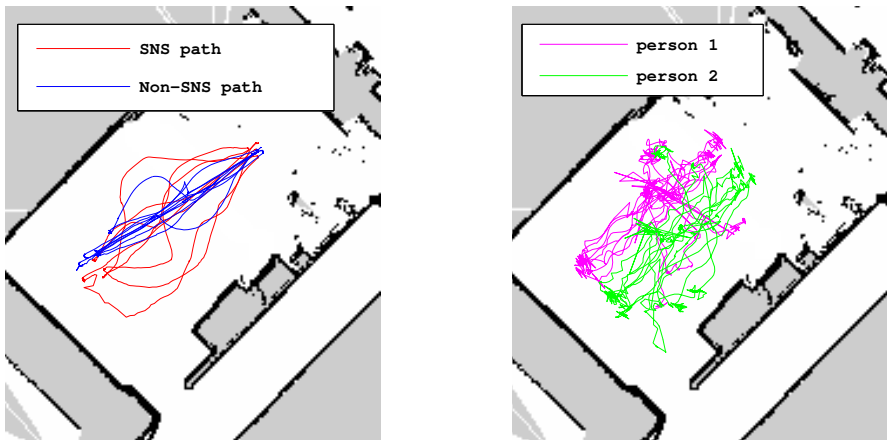


Figure 7.12: Representation of the paths followed in the dynamic free run experiment. Left: paths followed by the robot using the TERESA SNS (red lines) and MOVEBASE Non-SNS (blue lines). Right: Paths followed by the two people in the scenario.

able to behave in a socially acceptable way in an environment with moving pedestrians. To do that, the RRT* re-plans at a frequency of 2 Hz. In this scenario, two people are walking freely in the scenario, sometimes forming a group and sometimes walking alone. At the same time, the robot is receiving waypoints that lead it to cross the room avoiding the people around. One long run for each navigation system was performed. Figure 7.12 shows the paths followed by the people in one of the runs (right) and the paths followed by the robot with the TERESA SNS and the MOVEBASE Non-SNS (left). As can be seen, the SNS tried to deal with the people adapting its paths to the individuals in a social way unlike the Non-SNS, which tried to perform the same direct trajectory to the goal many times.

Table 7.5 shows the metrics obtained in the experiment. In this case, the total time T_p indicates the total duration of the run, being the recording of the SNS case a bit longer than the Non-SNS recording. Under this condition, the path length metric L_p indicates the total length traveled by the robot (all the trajectories), and, as can be seen, it was longer in the SNS case. That fits with the paths shown in Fig. 7.12 left, and explains the higher value in the cumulative heading changes (CHC) since the SNS tried to adapt its paths to the people in the scene. Again the most relevant

Table 7.5: Results for social navigation evaluation of a free run with two people moving in the scenario.

GROUP		TERESA	MOVEBASE
$T_p(s)$		495.50	**387.80**
$L_p(m)$		70.24	**50.18**
$CHC(rad)$		89.16	**74.34**
$CP_{avg}(m)$		2.50	2.50
$CP_{min}(m)$		**0.31**	0.27
$CP_{max}(m)$		4.65	**4.84**
$CP_{prox}(\%)$	Intimate	**0.60**	3.17
	Personal	14.31	**10.11**
	Social+	85.08	**86.73**
$IS_{prox}(\%)$	Intimate	**3.31**	8.51
	Personal	**12.27**	12.67
	Social+	**84.42**	78.81

and significant metrics are the intrusions in the Proxemics spaces of the closest person (CP_{prox}) and the interaction area (IS_{prox}). The intrusions in the intimate spaces are much lower in the case of SNS, what states the effort on minimizing the disturbance of people. As a general result, we can say that the behavior of the TERESA SNS is more socially acceptable than the behavior shown by the MOVEBASE Non-SNS as expected.

7.5 Integrated experiments

Throughout the TERESA project, four joint consortium studies were conducted. These consortium studies were aligned with four iterations of the robotic system and its software and included a longitudinal study and also new participants in each case. Some results about the acceptance of the system by the end-users along the iterations are presented here.

Then, the particular results regarding the last version of the robotic system (iteration 4) are presented. During these last experiments, all macro-actions from Section 7.2.3 except for the walking side-by-side were integrated and available for the visitor. Two different kinds of experiments are shown. The first one corresponds to a demonstration with the TERESA robot in an event called "Faite-vous Plaisir, Seniors"[19], about the well-being of elderly people, and that took place on October 19, 2016, at the Centre Sportif de l'Aube (Troyes, France). The second

[19] http://teresaproject.eu/wp-content/uploads/2016/07/TERESA-Dissemination-Press-release.pdf

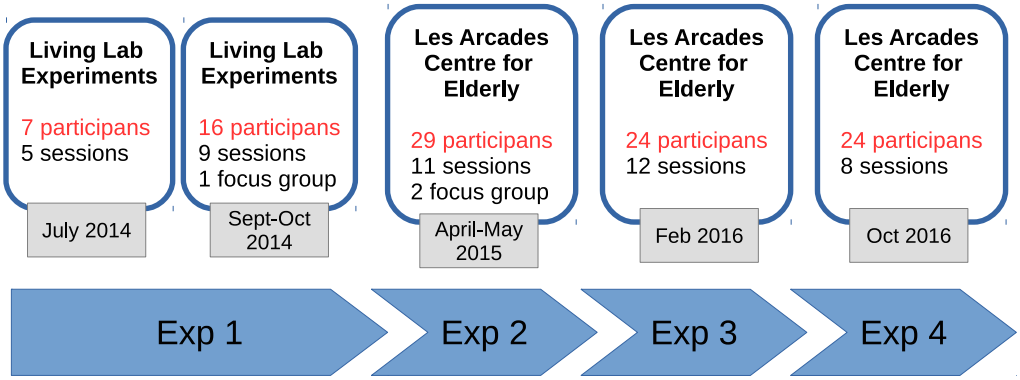


Figure 7.13: Overview of the TERESA consortium studies performed.

one corresponds to the experiment 4 of the project themselves, that took place at the day centre Les Arcades (Troyes, France), in October and November 2016.

7.5.1 End-user evaluations

Throughout the project, four joint consortium studies were conducted. These consortium studies were aligned with the integration of the four iterations of the robotic system and its software. Each of the consortium studies thus directly followed on the integration and deployment of a new version of the robot. That had as a downside that the protocols often needed minor adaptations dependent on the outcomes of the integration and deployment. The upside was that the protocols could be designed to fit the needs of the project at its different stages and could be adapted to suit the status of the different modules. This work was mainly conducted by Jered Vroon, from the University of Twente, and Raphael Koster, from the Expert Centre in Technologies and Services for the Maintenance and Autonomy of Elderly (Madopa), as partners of the project. The complete evaluation can be consulted in (TERESA Consortium, 2016b).

Between July 2014 and October 2016, 45 sessions with the robot were carried out over the course of four consortium studies (TERESA 1a in July 2014, TERESA 1b in September-October 2014, TERESA 2 in April-May 2015, TERESA 3 in February 2016, and TERESA 4 in October 2016, see Fig. 7.13). All in all, these sessions involved 100 participants (71 people), including ten members of the users' committee, who in nearly all cases had been present at every session right from the start of the project, and 61 new participants.

In the evaluations, some of the participants were acting as visitors (controlling the robot) and other acting as interaction targets. After experiencing the scenarios, participants were given a questionnaire about their experiences with

and attitudes toward the system. From those questionnaires, a percentage of acceptance in different aspects could be obtained. We will comment only the results that can be related to the robot social navigation, although it is not easy to isolate and evaluate this part from the general user experiences, as we discuss later.

It is important to emphasize that we are not comparing the same system between studies 1 to 3 and consortium study 4. That is, new functionalities were integrated for the first time. That means that some functionality went from mostly flawless Wizard-of-Oz-controlled movement (a hidden expert controlling the robot) to fully integrated autonomous behavior; while the system was more integrated, intelligent and autonomous, this also necessitated several changes to accommodate the different dependencies between all aspects of the system. Furthermore, not only the functionalities of the robot were different, but also the low-level electronics, the shell and look of the robot, and the interface had changed (See Fig. 7.14). For this reason, we discuss the results from consortium study four separately from those of consortium study 1-3. We do also discuss the relationship between these two versions with the intent of identifying the main ways in which robot motion was affected by these significant changes.

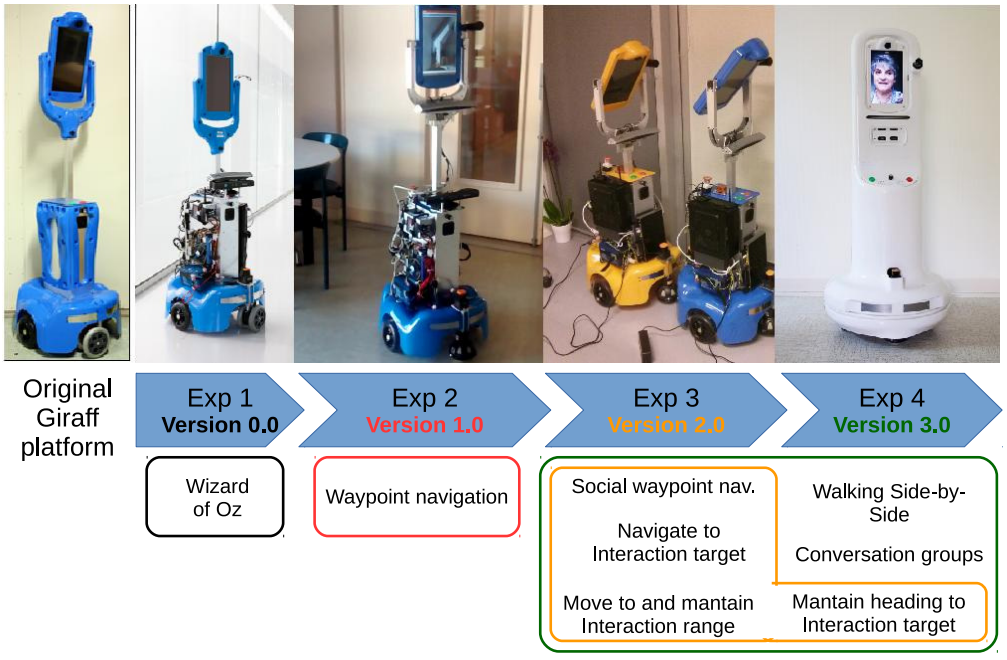


Figure 7.14: TERESA functionalities added to each version.

7.5.1.1 Study results

Here we summarize the acceptance of different aspects of the robot motion along the four consortium studies and therefore the four different versions of the telepresence robot. The aspects considered are grouped into these categories:

- **Technical skills and ease of use**. This category accounts for the technical difficulty that the user faced when handling the robot.

  In the first three consortium studies, the ten longitudinal participants (the group of people that participated in the four experiments) stated that the more autonomous the robot became, the easier it was to use for those controlling it and less technical knowledge was necessary. Despite greater autonomy of the robot in the version of experiment 3, the new participants were intimidated by the technical knowledge required instead.

  However, in the fourth consortium study, we found that most users were less enthusiastic. A completely new user interface, including a new system for manual control of the robot movement based on keyboard (previously was based on mouse), was employed. In contrast to what was found in informal prior evaluations of the new interface, the users had difficulty controlling the robot manually using the arrows on the keyboard. Despite that users could freely switch between manual and semi-autonomous control, the participants enjoyed the challenge of directly controlling the robot and felt a bit disappointed with it - regardless of how effective these modes of control were from a functional point of view. These issues should probably be resolved, as they could have a negative effect on the long-term use of the system.

- **Safety**. This category considers the reliability and safety that the users felt about the robot.

  In the first three consortium studies, the robot's autonomous behaviors seemed increasingly reliable and safe to the ten longitudinal participants. The rate of distrust among participants went down from 80% for the first two versions of the robot to 50% for version three. However, a bias in the protocol should be highlighted: version one of the robot simulated autonomy through a Wizard-of-Oz concealed from the participants. Dysfunctional behavior was caused deliberately so that reactions to it could be observed (study conducted by the University of Amsterdam). The major changes made to the robot's appearance in the fourth consortium study caused the participants of the longitudinal study to feel warier again.

  Regarding the group of new participants, up to version three, more than 50% of new users were wary of the robot and considered its movements

unreliable. There was, however, a slight improvement between version 1 and version 3.

The final social navigation system of the robot for consortium study 4 fulfilled its purpose: a greater feeling of safety and reliability appeared in new users. Only 27% of users in consortium study four still considered the robot to be unreliable, compared with 63% at the time of previous consortium study 3. The navigation was more reassuring because of the robot looks "medical" according to the opinion of the two groups (longitudinal and new participants).

- **Emotions: Stressful/Banal/Challenging/Surprising/Fun**. Some impressions and feelings about the robot appearance and behavior are considered in this category.

  Considering the participants of the longitudinal study, in the first three consortium studies we saw that the more autonomous the robot's movements became, the less stressful the control activity became. Interestingly, said stress was at the same time also a source of amusement: it nourished a sense of challenge in issues surrounding control of the robot, which gradually faded as the robot became more developed. The robot became less "fun", banaler and therefore also seemed less innovative.

  In the fourth consortium study, the difficulties mentioned above in manually controlling the system caused considerable stress. Fun was comparable to that in the third iteration, which may be caused either by this stress or by the fact that the robot was comparably autonomous.

  Regarding the group of new participants, paradoxically, the more autonomous the robot's movements became, the less amazing it seemed. Its innovative nature was more perceptible when users were given the power to control the robot themselves. The less the participants controlled the robot, the more they felt stressed about not being behind the controls.

  That held true for the semi-autonomous robot in the final session of consortium study 4, whose fun aspect had completely disappeared. That could be related to the aforementioned paradox, making the robot less fun as it becomes more autonomous.

7.5.1.2 Introspection and conclusions

As previously commented, for the final consortium study in October 2016, the TERESA robot's interface and appearance were thoroughly overhauled. The changes made between the old and new versions of the robot are so major that it is worthwhile comparing the acceptability of the old version (TERESA1-3) with

that of the new (TERESA 4). Changes to the robot's interface, its appearance, its hardware, and its software all had a direct impact on the robot's acceptability for users.

While taking into account that comparing these versions is difficult precisely because of these differences, we try to identify here the main ways in which robot motion was affected by these changes.

When comparing the acceptability of the old and new versions of the robot, an important outcome is that, with the version of the robot used in the last consortium study (the one with better autonomous capacities), participants indicate that they want to retain access to and control of the robot.

That seems a paradoxical and unexpected finding for a project aimed at developing the autonomy of a telepresence robot. Instead of focusing on the interaction with the people in the scene while the robot is autonomously navigating or accompanying someone, the users (elderly) feel the necessity of controlling the robot, even if that penalizes the attention devoted to the interaction. The reason exposed was that they had the feeling of being dominated by technology.

For these reasons and considering the previous results, the general evaluation of a social robot is complex. A lot of different component needs to work together, and small details in one of them can have a direct impact on the perception of the others by the users. Thus, the isolated evaluation of the different components of the full system becomes difficult to determine. In this sense, concluding about the acceptability of the "social" autonomous navigation cannot be fairly done.

7.5.2 Centre Sportif de l'Aube

As commented, the robot was demonstrated during an event about the well-being of elderly. The place of the event consisted of an area of 25 x 28 m approximately counting with a main hall with two corridors and stairs, and two rooms (Fig. 7.1 shows the robot at the main hall). Several booths of different companies were there.

During the demo, the robot TERESA could be used by any visitor. The autonomous navigation and conversation capabilities were activated. Many people were visiting the different stands walking around or standing conversing while having a coffee which turns the environment challenging for navigation.

The robot was working successfully the whole day in that environment, from 9.30 to 17.00. Figure 7.15 shows some instants of one navigation through the environment. A video of the experiment can be accessed at the D4.3: Navigation Demonstrator webpage[20].

[20] `http://robotics.upo.es/teresa/teresa-d43.html`

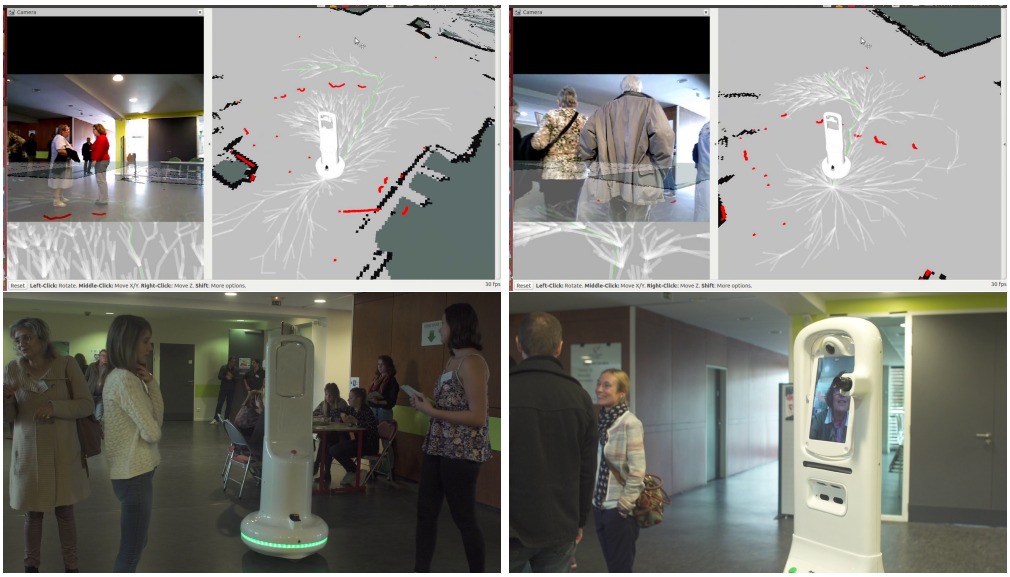


Figure 7.15: Some instants of different trials at the Centre Sportif. Top: "robot view". Bottom: some pictures of the demo.

7.5.3 Les Arcades

The final experiments of the project took place at the elderly day centre "Les Arcades" in Troyes, France, from October 24^{th} to November 2^{nd} 2016. Figure 7.16 shows some pictures of people interacting with the TERESA robot at Les Arcades. A full description of the experiments can be found in the Deliverable D6.6 (TERESA Consortium, 2016b).

The different navigation macro-actions (except the walking side by side) were employed during Experiment 4. Figure 7.17 shows an illustrative example of the approaching macro-action where the robot performs an approach to the indicated person by the user to initiate an interaction. Another example, in this case for navigation to waypoint macro-action, can be seen in Figure 7.18.

The videos of many sessions of experiment 4 can be consulted in the D4.3: Navigation Demonstrator webpage where the behavior and transitions between the different modules of the integrated system can be observed. Explanations about the information represented in the videos are also provided.

7.6 Conclusions

This Chapter presented the integration of some of the techniques developed in this thesis in the real robot of the European project TERESA (TElepresence

Figure 7.16: Pictures of Experiment 4 at Les Arcades.

REinforcement learning Social Agent). We focused on the navigation system developed for the project, which has been augmented with different social navigation skills. The system and relations between the different modules involved were described. Moreover, the different experiments and evaluations performed in real environments are presented.

The navigation stack considers global planning, local planning, and execution. Normative social behavior was included in cost functions, and social constraints learned from demonstrations. Thus, the novel learning algorithm presented in Chapter 6 has been applied to endow the TERESA robot with social navigation capabilities. That has been compared to non-human aware stacks and shown how it led to different motion behaviors.

Furthermore, the learning framework presented in Chapter 5 was also employed for the task of approaching a person to interact with socially. The adaptation and use of the TERESA robot were explained. Also, the integration of the learned behavior with the other available actions in the system was described.

The navigation stack has been successfully deployed and tested in real environments. The results of controlled experiments and evaluations of the system were presented here. Moreover, we showed the tests performed at the Centre Sportif de lAube, in Troyes, France, during a one-day event (uncontrolled scenario). It also includes videos of the tests in the elderly care center Les Arcades, where the robot was working with the navigation stack during several trials for two weeks.

Through the experimentation with the real robot in real environments, we stated the importance of a reliable perception system for the surrounding people. However, this a problem not fully accomplished yet. Correct detection and tracking of people positions and orientations, or even head or gaze direction, is crucial for a correct understanding of the social situation. Light conditions, robot movement, partial occlusions and many other factors concern the detection problem. Better detection algorithms and more advanced data fusion from different sensors are required. In our case, false positive detections (some examples can be seen in Fig. 7.17 and 7.18) led the robot to show an unusual behavior in some particular cases.

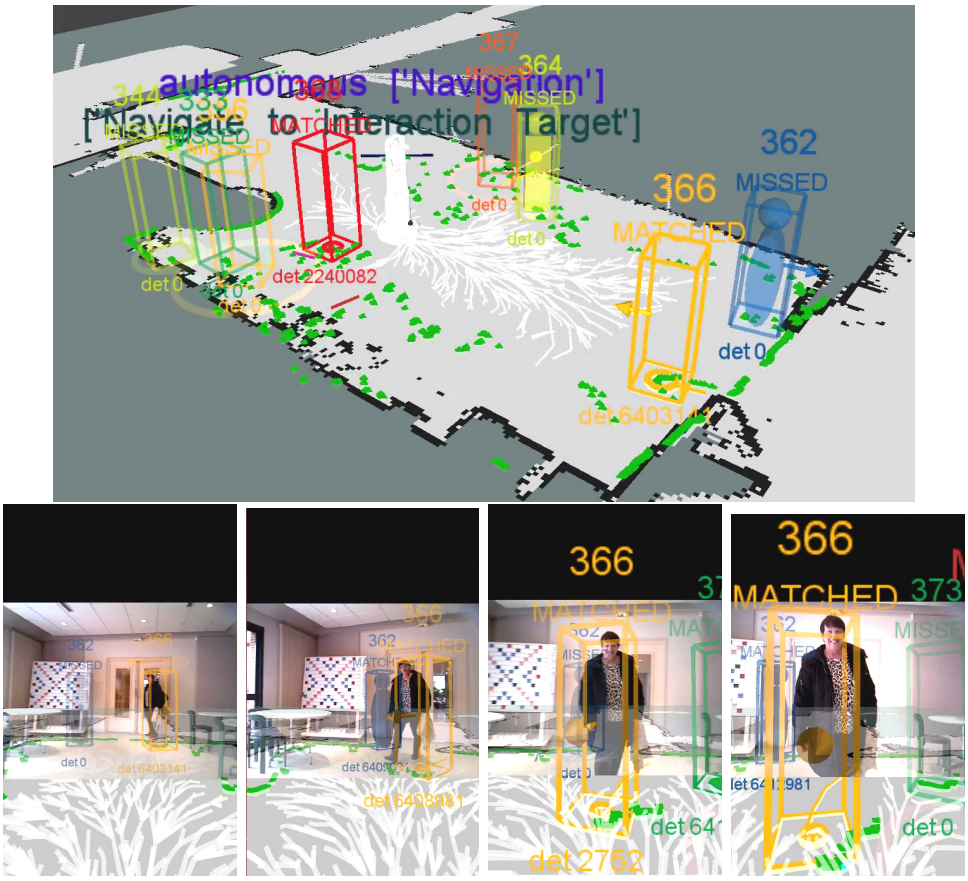


Figure 7.17: Approaching an interaction target at Les Arcades. The robot is commanded, through the interface, to approach the person with identifier 366 (in the interface, the user just have to click on an oval that appears on the person).

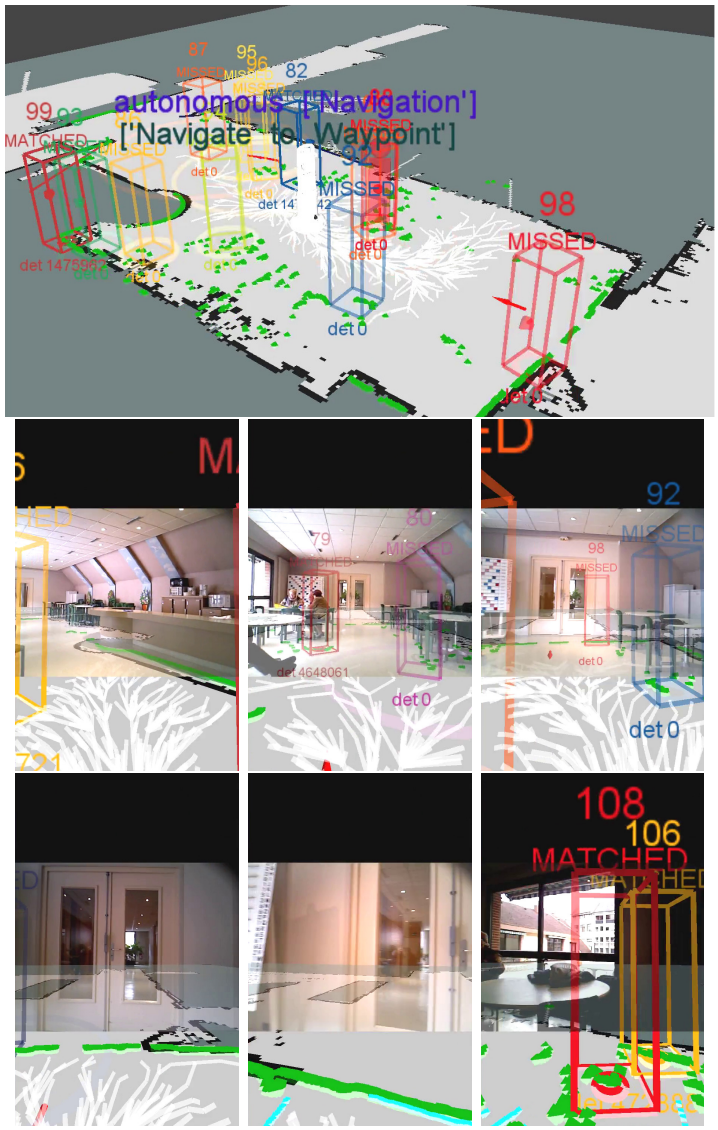


Figure 7.18: Navigation to a waypoint at Les Arcades. The robot is commanded, through the interface, to go to a certain predefined waypoint (in the interface, the user just have to click on the name of the waypoint where she/he is interested to go).

Chapter 8

Fully Convolutional Networks to learn human-aware path planning

"They're here!"
Poltergeist, 1982

This Chapter presents an approach to learn path planning for robot social navigation by demonstration from a different perspective than those presented in previous chapters. We make use of Fully Convolutional Neural Networks (FCNs) to learn from expert's path demonstrations a map that marks a feasible path to the goal as a classification problem. The use of FCNs allows us to overcome the problem of manually designing/identifying the relevant features for the task of robot navigation, as we identified in Chapters 3 and 6. The method makes use of optimal Rapidly-exploring Random Tree planner (RRT*) to overcome eventual errors in the path prediction; the FCNs prediction is used as cost-map and also to partially bias the sampling of the configuration space, leading the planner to behave similarly to the learned expert behavior. The approach is evaluated in experiments with real trajectories and compared with Inverse Reinforcement Learning algorithms that use RRT* as the underlying planner.

8.1 Introduction

In Chapter 6, we presented an IRL approximation for learning human-aware navigation behaviors. It uses a RRT* planner instead of an MDP to overcome some of the computational drawbacks of the MDPs (Pérez-Higueras et al., 2017). Besides that, most of the IRL techniques present other issues when the task to be learned is complex, like the human-aware navigation. In these cases, we have to manually design the structure of the reward and features involved in the task. In many cases, the designed reward probably is not able to recover the underlying demonstrated behavior properly. Even though the weights of the

cost function can be learned from expert demonstrations, we still need to define a set of features functions that describe the state space and the relation between them.

However, in the last years, the use of deep neural networks in IRL problems is bringing good results. Neural networks, as function approximators, permit the representation of nonlinear reward structures in domains with high dimensionality. Some authors have already applied deep networks to the problem of human-aware robot navigation among others. In (Chen et al., 2017a), a Reinforcement Learning approach is applied to develop a socially-aware collision avoidance system where a deep network is employed to learn multi-agent crossing trajectories. Also, Deep Q-networks have been used in (Sharifzadeh et al., 2016), to solve the MDP step of an IRL algorithm for the task of driving a car. Furthermore, the well-known Maximum Entropy IRL (Ziebart et al., 2008) algorithm is extended in (Wulfmeier et al., 2015) to use deep networks, and its application to path planning in urban environments is presented in (Wulfmeier et al., 2016). Another example is (Shiarlis et al., 2017b) in which a deep network is employed to learn from demonstration a robot control policy for the task of keeping the correct distance and orientation in a conversation with humans.

In this Chapter, we present a different approach to those commented above and the ones presented in Chapters 5 and 6. Here, we propose a learning from demonstration strategy for the task of human-aware path planning that avoids the explicit representation and definition of the cost function and its associated features. The proposed method does not follow the classical IRL approach. Instead, the problem is formulated as a classification task where a Fully Convolutional Network (FCN) is used to learn to plan a path to the goal in the local area of the robot in a supervised way, using the demonstrations as labels. This general approach is then combined with an optimal path planner to solve the task efficiently and to ensure collision-free paths, similarly to the sampling bias performed in Chapter 5. Thus, the contribution presented in this chapter is twofold: 1) a novel approach for learning human-aware path planning from demonstrations based on Fully Convolutional Networks and 2) the combination of this information with an RRT* planner to enhance the planner capabilities while it behaves similarly to the expert behavior.

8.2 Learning to plan paths with FCNs

In this section, we present our approach to learning from demonstrations to plan human-aware paths in 2D environments for robot social navigation. We propose using a Fully Convolutional Network to learn, from the expert's path demonstrations, to predict the correct path to the goal according to the current

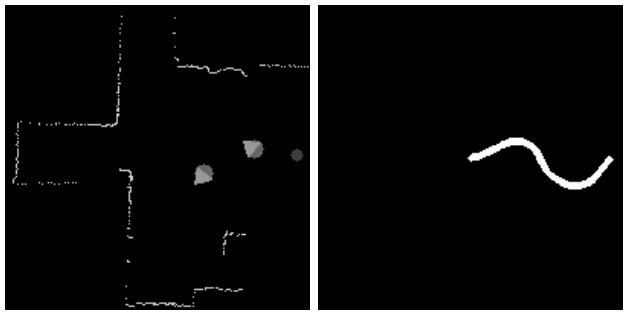


Figure 8.1: Left. Example of input grid to the network. Obstacles, goal position, and people positions and orientations are included. The robot is always located in the center of the image and looking to the right direction, so the rest of elements are properly oriented around the robot. Right. Respective label used for learning with the path to the goal.

obstacles and people in the vicinity of the robot, without explicitly defining the environment features used. Unlike Deep-IRL approaches, which try to learn the cost function that the expert is following in an MDP framework, our goal is to directly predict the path to the goal given the information from obstacles and persons.

The path predicted by the FCN is then combined with a RRT* planner to perform the navigation task efficiently and to prevent the robot from collisions or prediction fails.

8.2.1 Input and output of the network

We consider a local navigation task in a 2D space of size $10 \times 10\ m^2$, centered on the robot, which is always in the center and oriented to the right. Thus, the sensor data are transformed according to the robot local coordinate frame and used as input to the network. No other information is employed, so the network has to derive the features of the task based on the provided state information.

The sensor data based on laser scans and point-clouds is projected in a 2D grid of $200 \times 200\ pixels$ with a resolution of $0.05\ m/pixel$. As we are considering the task of social navigation, we need to be able to detect people. So, for real conditions, the same people detection and tracking system used in TERESA (Chapter 7) is employed here to provide information about the position and approximated orientation regarding the robot position and orientation. This data is also included in the 2D grid, where the people are marked using circles and triangles to indicate the orientation. The goal is also marked on the grid as a small circle. No prior information about the area, like global maps, is employed

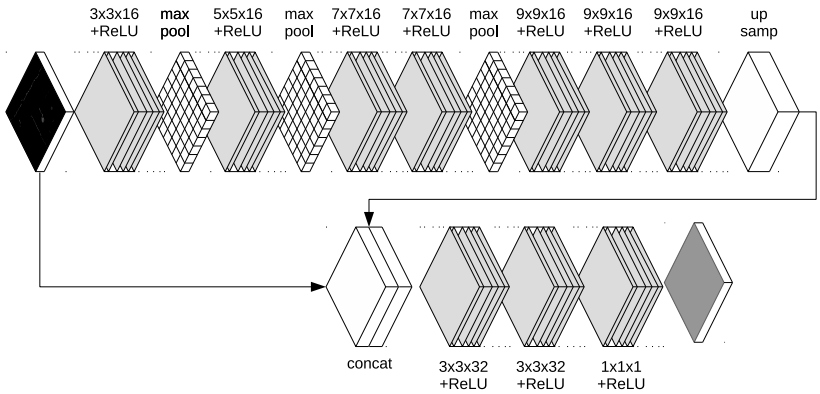


Figure 8.2: Proposed network architecture. The network is composed by two branches, a global-coarse branch that extracts high-level features of the input grid and a local-fine branch that make use of these features and the original input grid to generate the final path.

in this case, but it could also be combined easily. This grid is used as a gray-scale image where the background color is black and each element previously described takes a different gray intensity. These values are normalized to be in the range $[0, 1]$ before serving as input to the FCN. An example of the input gray-scale image of the network can be seen on the left image of the Fig. 8.1.

The output of the network is an approximated path from the center (robot position) to the goal marked on a grid, which has the same size as the input. We can see an example on the right image of the Fig. 8.1.

8.2.2 Network architecture

Path planning problems are especially characterized by the existence of global information in its formulation, as a goal destination among others. That means the points of the path are not only affected by local information, but also from global information. This information is critical to optimize the path searching policies. For instance, the robot needs to know where to go before plan the best trajectory from its current position.

The importance of the global information has been considered into the design of the proposed neural network architecture, presented in Fig. 8.2. The architecture is inspired by the global-coarse to local-fine deep learning architecture presented by (Eigen et al., 2014). Thus, the proposed structure is divided into two major branches: a global-coarse estimation that sequentially subsample the input grid while applying larger kernels in order to extract global high level features of the input, and the local-fine branch that make use of the extracted

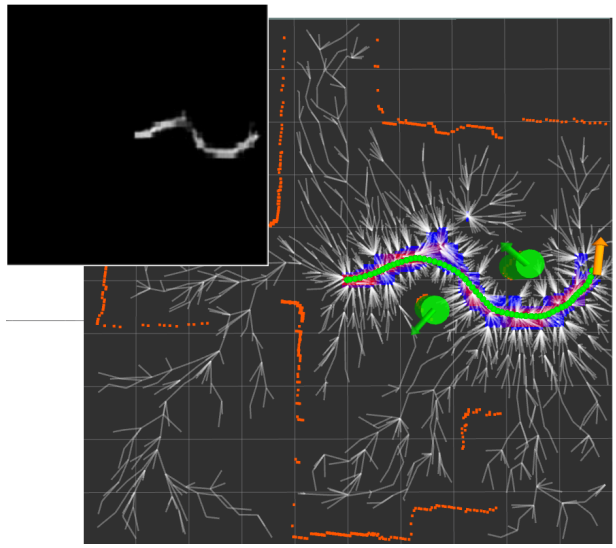


Figure 8.3: Upper left. Example of output plan of the network. Down right. RRT* path using the network prediction with a bias of the sampling of 70 %. Robot position is the center of the image. Orange lines show the obstacles. People position and orientation are indicated with green cylinders and arrows. The goal is represented by the golden arrow. The planner tree is drawn in white color.

global features and the original input grid to build the final path considering local information and global constraints.

Notice how the proposed network does not make use of fully connected layers, which significantly reduces the total number of parameters to one hundred thousand approximately. Instead, an output layer with 1x1 kernel size is used to generate the output grid with the path.

8.2.3 Integration with the RRT* planner

The final step consists in combining the path predicted by the FCN with a RRT* planner to perform the navigation task efficiently and to prevent the robot for collisions or prediction fails.

The planner uses the path prediction following the same idea that we employed in Chapter 5: first, as a cost function to connect the tree nodes; and second, to partially bias the sampling of the state space, taking a higher percentage of samples from the areas where the predicted plan is. That allows to guide the sampling process efficiently to areas of interest and thus, reaching an optimal path faster.

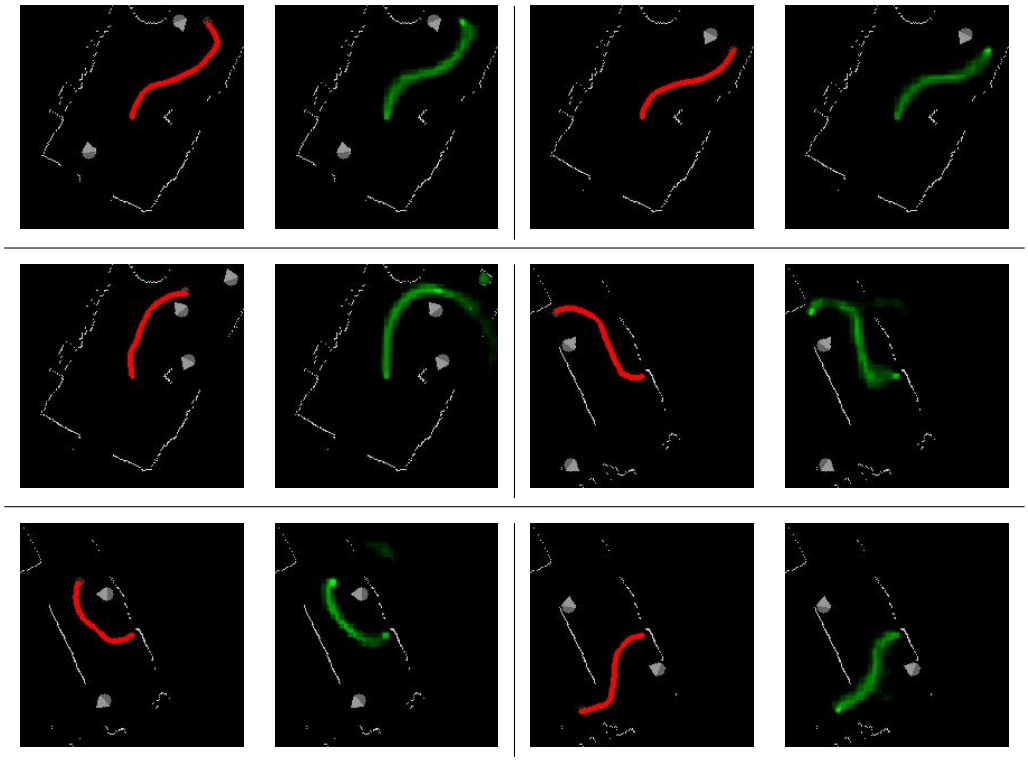


Figure 8.4: Visual comparison of 6 pairs of labels and network predictions from the testing dataset. For each pair, the label is represented in red line on the left image, and the prediction is shown in green color on the right image.

The path prediction from the network and the resulting RRT* path for the same example presented in Fig. 8.1 are shown in Fig. 8.3. As can be seen, the resulting path is very similar to the demonstrated one.

The percentage of samples drawn from the path prediction or a uniform distribution is an important parameter. Keeping a percentage of uniform sampling allows the planner to be still able to find a path to the goal when the prediction is not complete or fails. In this case, we have used a 70 % of samples from the path prediction.

8.3 Deep Network Validation

We first validate the deep learning approach by testing the performance of the network prediction. A dataset of 10500 trajectories has been used to validate the capability of the proposed FCN to learn to plan paths. The dataset has been

Table 8.1: Mean Squared Error (MSE) reached in the different stages with different set of trajectories

	Learning set	Validation set	Testing set
MSE	0.0073	0.0067	0.0094

created by randomly generating positions and orientations for the robot, the goal and different number of people around in different scenarios within a large map.

Regarding the generation of demonstration paths to the goals in the scenarios, a RRT* planner with a pre-defined cost function has been employed. In particular, as a cost function, we employ the same weighted linear combination of five features that we used in Chapter 6 for robot social navigation. Recall that these features are based on distance metrics as the Euclidean distance to the goal and the closest obstacle. Also, metrics regarding the people in the vicinity of the robot are taken into account. The Euclidean distances and orientations concerning the people in the scene are used through three Proxemics functions placed in front, back and sides of each person.

From the dataset, 10000 trajectories are used in the learning process, while the remaining 500 trajectories are reserved for testing the network after learning finishes. Also, during the learning, 100 trajectories were taken from the learning set as a validation set for overfitting checking. The results in terms of Mean Squared Error (MSE) between the pixel values of the network output and the respective image label for the different sets of trajectories are presented in Table 8.1. As can be seen, the error committed in the testing set keeps low regarding the error reached in the learning set.

Furthermore, a visual comparison of some network predictions corresponding to some of the 500 trajectories of the testing set is presented in Fig. 8.4. The images are presented in pairs with the demonstrated path in red in the left image, and the respective prediction on the right image in green. As can be observed, the predictions fit the expert's paths very well.

An interesting and unexpected outcome of the proposed network is its capability to generate more than one valid path when the goal can be similarly reached by following different homotopies, even when the learning has been made with a single trajectory per example. Figure 8.5 shows two examples of prediction in two valid homotopies. This information is easily handled by the RRT* planner which will select the shorter path.

The source code of the network and all the datasets used in the experiments

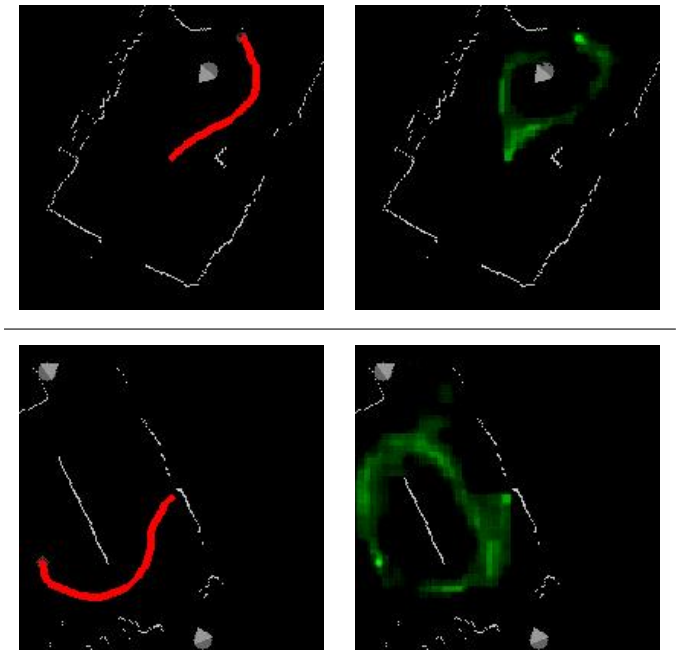


Figure 8.5: Example of two cases where the network learn to plan in different valid homotopies. The demonstrated path is shown in red color on the left images while the respective prediction is shown in green color on the right side images.

can be found in the Github repository of the UPO Service Robotics Lab[1].

8.4 Path Planning Evaluation

The feasibility of the proposed approach is demonstrated for human-aware planning by learning in a small dataset of real robot trajectories. Then we compare the resulting trajectories with a ground-truth set and two IRL algorithms of the state of the art that learn the cost function of a RRT* planner as a weighted linear combination of features.

8.4.1 Dataset

A dataset of 300 trajectories has been recorded in different simulated environments with static people around and where the robot was controlled by a human expert who tried to perform an accurate human-aware navigation to the goal.

[1] `https://github.com/robotics-upo/upo_fcn_learning`

This dataset is sorted in 3 sets, where 250 demonstration paths are used for learning, and 50 are used for testing the results. The difference between sets is that the 50 paths chosen for testing (and thus the 250 for demonstration) are different for each set.

Is noteworthy that, regarding deep learning with FCNs, a dataset of 250 demonstrations for learning human-aware navigation can be considered very small. We will show that even with this small information the presented approach can equal or overcome the result of two state-or-the-art IRL algorithms.

8.4.2 State-of-the-art algorithms

The performance of our approach is tested against two IRL algorithms of state of the art: RTIRL (Pérez-Higueras et al., 2017) and RLT (Shiarlis et al., 2017a). Both try to learn the weights of a linear combination of features used as cost function of a RRT* planner using the same set of demonstrated trajectories. The first one is presented in Chapter 6 of this thesis and is based on a Maximum Entropy approach (Ziebart et al., 2008) that replace the MDP by an RRT* planner. The second one is an adaptation of the Maximum Margin Planning algorithm (MMP) (Ratliff et al., 2006) to work with a RRT* planner.

We have used with these algorithms the extended set of features specifically designed for robot social navigation of Section 6.3. The set is a compound with five features. Three of them are based on distances and orientations between the robot and the people in the scene and are coded through three Gaussian functions placed in front, back and side of each person in the scene. Then, two more features measuring the distance to the goal and distance to the closest obstacle are also considered. It is important to remark that our approach, on the contrary, is fed with just the raw laser data and the position and orientation of people in the form of an image, as explained in Section 8.2.1.

The implementation of the IRL algorithms used for comparison can be found in the module *upo_nav_irl*[2] of the Github repository from the UPO Service Robotics Lab.

8.4.3 Metrics

In order to compare the planned paths with the ground-truth paths, we use the same Trajectory Difference Metric (TDM) that we employed in Chapter 5 (Eq. 5.8), which is based on a directed distance measure between two paths:

$$TDM(\zeta_1, \zeta_2) = \frac{\sum_{i=1}^{N} d(\zeta_1(i), \zeta_2)}{N} \tag{8.1}$$

[2] `https://github.com/robotics-upo/upo_nav_irl`

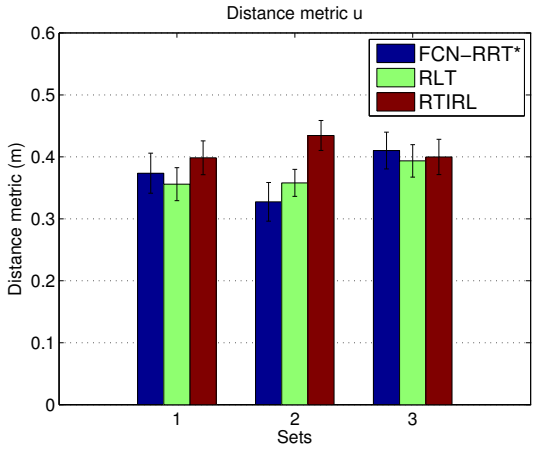


Figure 8.6: Average distance metric μ and standard errors for the three approaches in the three sets of trajectories of test.

where the function $d(\zeta_1(i), \zeta_2)$ calculates the Euclidean distance between the point i of path ζ_1 and its closest point on the path ζ_2, and N stands for the number of points of the path ζ_1. This distance in then combined to obtain a final metric of path comparison μ:

$$\mu(\zeta_1, \zeta_2) = \frac{TDM(\zeta_1, \zeta_2) + TDM(\zeta_2, \zeta_1)}{2} \tag{8.2}$$

Moreover, a comparison between the difference in the feature counts of the ground-truth paths and the planned paths is also employed, as the feature counts play a key role in the IRL approaches (Pérez-Higueras et al., 2017; Shiarlis et al., 2017a). According to the path cost calculation presented in Eq. (6.2) of the chapter 6, the calculation of the feature count of a path is defined as:

$$F(\zeta) = \sum_{i=1}^{N-1} \frac{f(\zeta(i)) + f(\zeta(i+1))}{2} \|\zeta(i+1) - \zeta(i)\| \tag{8.3}$$

where $f(\zeta(i))$ indicates the vector of features values for point i of path ζ, and $\|\zeta(i+1) - \zeta(i)\|$ is the Euclidean distance between the points i and $i+1$ of the path ζ. Even though our FCN approach does not make use of such features, we also compute their counts for the resultant trajectories.

8.4.4 Comparative results

We first train the RLT, RTIRL, and FCN approaches with the learning set, and then compare the results using the testing set. We obtain the results for the

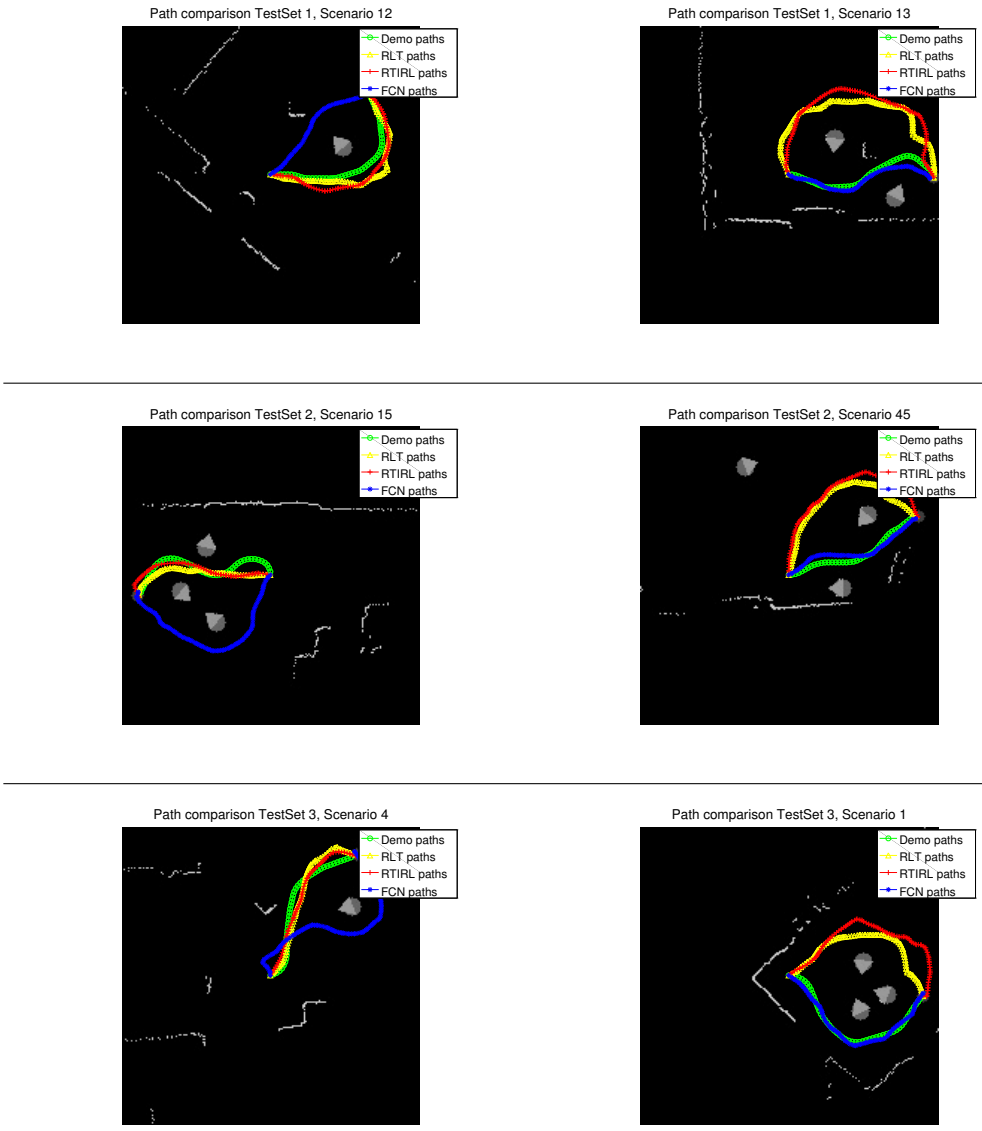


Figure 8.7: Visual comparison of paths in some scenarios for each testing set, in which the FCN-RRT* approach fails (left side images) or fits the demonstrations paths (right side images).

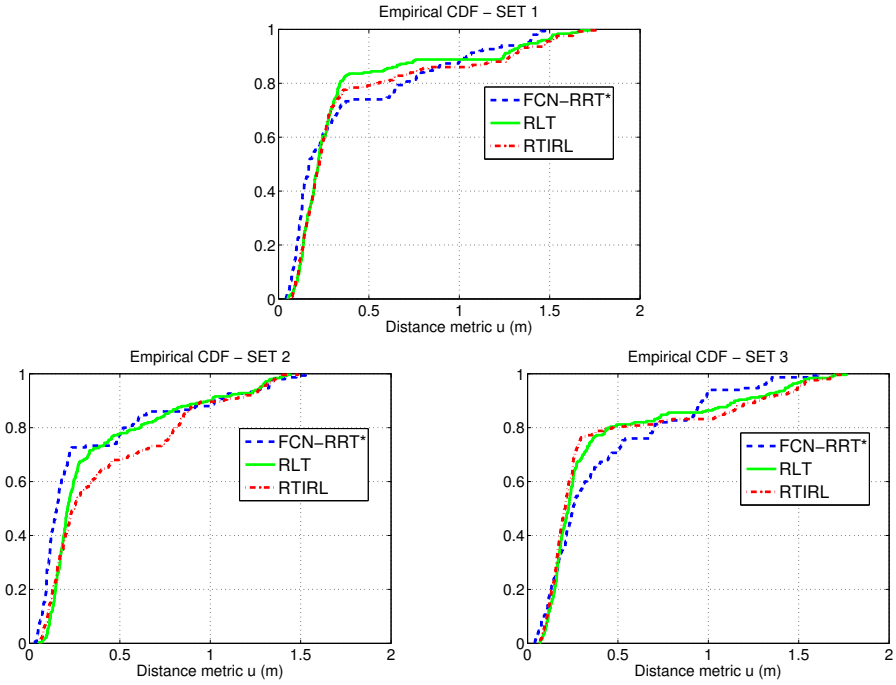


Figure 8.8: Cumulative error in the distance metric μ for the different approaches in the three sets of trajectories for testing.

three different data combinations described above.

First, the average values of the distance metric (Eq. 8.2) between the planned paths and the 50 ground truth trajectories of each testing set are shown in Fig. 8.6. As can be seen, the performance of the three approaches is very similar. Even though no further information is provided to the FCN approach, it can learn an adequate representation of the task and equals the results obtained by the other two algorithms that use a pre-defined set of features engineered for human-aware robot navigation.

Moreover, a visual comparison of the paths for two examples of each testing set can be seen in Fig. 8.7. For each testing set, a scenario in which the FCN-RRT* plans in a different homotopy from the demonstration is showed on the left side. On the right, it is shown a scene where the proposed approach fits the demonstration, and the other algorithms fail.

Figure 8.8 shows a comparison of the cumulative density functions of the distance metric μ for the different approaches in the three sets of trajectories for testing. Again, the performance of our approach was similar to the other approaches without explicit information of the navigation features.

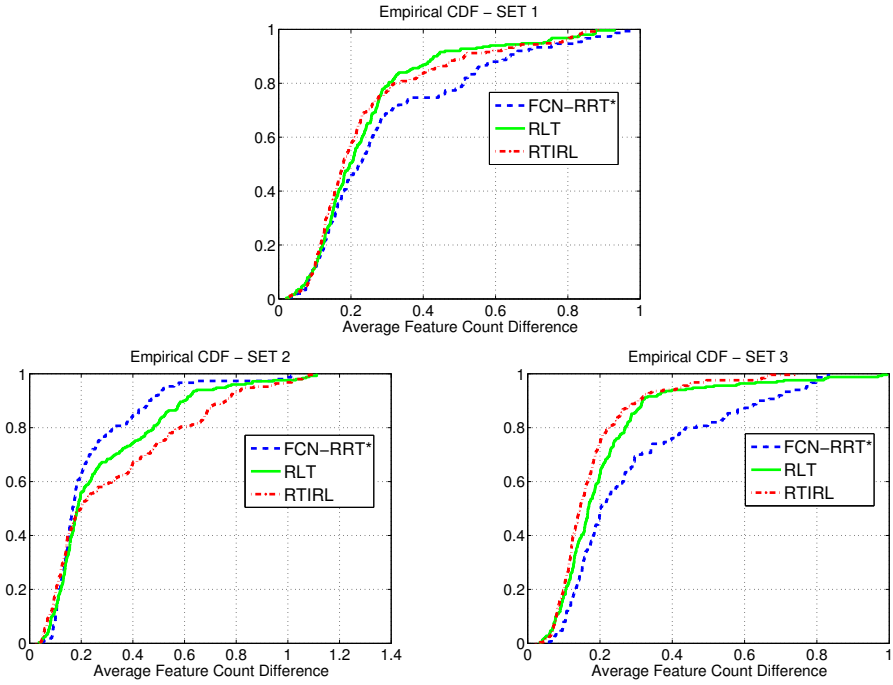


Figure 8.9: Cumulative error in the average feature count difference for the 3 approaches in the 3 sets of trajectories for testing.

Finally, we make a comparison regarding the manually-designed features. We compare the differences in the feature counts of the planned paths concerning the ground-truth paths of the three testing sets. Figure 8.9 shows the cumulative density functions of the average feature count difference. It is interesting to see how the RTIRL and RLT obtain similar results while our method underscores in two of the sets, but the resulting paths are very similar according to Fig. 8.8. That is an expected result given that the navigation features have not been specified to our approach, so the FCN could likely converge to another set of features that also leads to a good imitation of the demonstration trajectories.

8.5 Conclusions

This Chapter presents an approach for learning human-aware path planning from demonstrations in static scenarios based on the integration of FCNs and RRT*. The introduction of FCNs to learn the path planning based on demonstration allows us to address the problem without the need of hand-crafted social navigation features as many other IRL approaches in the state of the art, like those

presented in Chapters 3 and 6. Additionally, the integration of the predicted path with RRT* guarantees an optimal feasible path no matter the situation, similarly to the partial bias of the space sampling presented in Chapter 5.

The full approach has been tested with a set of real trajectories and benchmarked with state-of-the-art algorithms for human-aware navigation learning with good results. Our approach offers very similar metrics without defining the navigation features. Also, the prediction of plans in different and equally valid homotopies enhances the flexibility of the approach. Besides, the neural network prediction has also been successfully tested with a large dataset to validate the predicted paths.

Definitely, FCNs are a promising tool to capture the underlying behavior showed in the demonstrations. They allow us to overcome some of the problems encountered in the other approaches presented in this thesis. A further comparison with different network architectures and mainly, learning from demonstration in dynamic scenarios should be taken into account.

Chapter 9

Conclusions and Future Work

"Hasta la vista, baby."
Terminator 2: Judgement Day, 1991

Human-aware robot navigation is a hard problem due to the high complexity of the interactions that take place in populated environments. In this thesis, we have presented methods to enable mobile robots navigating in a socially compliant way, and that improve the behavior of traditional motion planning methods in social scenarios.

Instead of trying to describe and mathematically define all those interactions, we took a different approach. We studied and extended the capabilities of Learning from Demonstration (LfD) techniques to acquire the underlying social constraints from human examples of good behaviors and also to transfer these behaviors to unseen scenarios.

9.1 Discussion

In Chapter 3, we proposed the first approximation to human-aware navigation based on Inverse Reinforcement Learning (IRL). The fundamental concept was to learn from human demonstrations some basic social navigation behavior at a reactive motion level. Two simple discrete-MDP models (and combinations of them) to capture the behavior were presented. A public annotated dataset of real pedestrians was employed to extract the demonstration for learning. Then, the control policies obtained by solving the MDP models were compared with the human trajectories and a policy based on Proxemics theory. The results indicated that the proposed approach improves the Proxemics-based policy and can be used to transfer partial human navigation behavior into the low-level navigation controller of a mobile robot. However, these results can be improved. The computational complexity to solve MDPs led us to a gross discretization of the feature and control spaces to keep the MDP computationally tractable.

It turned out that the learned policies were not able to retrieve all the key-insight involved in social navigation. Therefore, a richer set of features, including velocity-based features, and more complex models can be considered.

The reward cost functions learned with the technique presented in Chapter 3, were employed to extend a robot local motion planner for obstacle avoidance with social skills. It was successfully integrated into the robotic tour guide of the European Project FROG, as shown in Chapter 4. On the one hand, the approach was satisfactorily compared with a non-social approach and a Proxemics-based approach. Moreover, the experiments and evaluations performed with the real robot in the project scenarios stated the success of the robot to perform robust and partially social navigation in very challenging and crowded environments. On the other hand, the "social" skills acquired were insufficient, as commented previously, and the kinematic constraints of the FROG robot did not allow it to have the required quick and agile movements according to the surrounding pedestrians.

The limitations imposed by the MDPs and the regular reactive motion planners applied to social navigation, as observed in the previous Chapters, motivated us to use a more flexible planner, like RRT*, for human-aware navigation in a local area. The use of RRT* also allows to work in continuous spaces and scales better than MDPs with larger state spaces and higher dimensionality. Thus, a different framework for modeling human-robot interactions based on imitation learning and sampling-based planners for execution was presented in Chapter 5. The approach made use of Gaussian Mixture Models (GMMs) to statistically model the physical interaction of the robot and the person when the robot is teleoperated by an expert demonstrating a task. Then, the learned models were integrated into a RRT* sampling-based planner at two levels: guiding the sampling step of the state space and including a cost term of the cost function employed by the RRT*. The idea was to exploit the benefits of the GMM to characterize the demonstrated tasks and the flexibility of the RRT* to overcome unseen situations. The approach was successfully tested in two particular navigation tasks: avoiding a standing person from two specific angles and approaching a person. However, this approach is very focused to mimic the demonstrations provided, unlike the IRL method that tries to learn the underlying demonstrated behavior. Therefore, it is difficult to apply the proposed approach to the general problem of human-aware navigation which involves several complex interactions and variability.

The low flexibility of the previous approach to generalize to unseen scenarios and to cover the diverse situations that arise in navigation motivated us to explore the possibilities of the IRL techniques again. We devised a novel learning algorithm combining the Inverse Reinforcement Learning (IRL) concepts and a

RRT* planner, as presented in Chapter 6. The aim was to learn from expert's path demonstrations the weights of the RRT*'s cost function that leads the planner to behave similarly to the demonstrated examples. The proposed method is simple to implement and allows to overcome the commented problems associated with IRL based on discrete-MDPs. Furthermore, the use of the RRT* permits to deal with larger scenarios (regarding people and features) than previous discrete-MDP-based approaches, like the ones presented in Chapter 3. The approach was validated against a ground-truth cost function and two related algorithms from the state-of-the-art. The results showed that the approach was able to approximate the demonstrated cost function and to obtain behaviors more similar to the demonstrations than similar state-of-the-art algorithms. However, the proposed approach can be extended in different directions: firstly, the learning considers only static scenarios, so a richer set of features, including the velocities of people and the robot beside considering the kinodynamic constraints into the planner, could improve the behavior in dynamic scenarios. Secondly, the identification of the relevant features involved in the social interaction is not clear, so other techniques to learn also the features from the demonstrations could be considered.

This latter approach was employed to learn the weights of the cost function employed in the social navigation system of the telepresence robot of the European TERESA project. Also, the unsupervised learning approach presented in Chapter 5 was satisfactorily applied to the task of approaching a human interaction target. The successful integration of these approaches and the experiments and evaluations performed encompassed in the TERESA project were presented in Chapter 7. The comparison of the TERESA navigation system with non-social approaches stated that the system was able to acquire some social skills. The TERESA platform was also lighter and agiler than the FROG robot, what eased the movement of the robot and thus, better social interaction was achieved. However, a holonomic platform could even improve the movement in navigation.

Finally, in Chapter 8, we addressed the issue of manually designing/identifying the cost-map and the relevant features involved in the task of human-aware path planning. To overcome this, we made use of Fully Convolutional Neural Networks (FCNs) to learn from expert's path demonstrations a map that marks a doable path to the goal as a classification problem. Then, the FCN path prediction was used as cost-map and also to partially bias the sampling of the configuration space of a RRT* planner, leading the planner to navigation plans similar to the demonstrated ones. The use of FCNs allowed to face the learning problem without the need of hand-crafted features as many other approaches in the state-of-the-art. The approach was tested with a set of real trajectories and benchmarked with a state-of-the-art algorithm and the IRL algorithm proposed in Chapter 6. The results indicated that the approach was able to equal the

performance of these algorithms without the need of defining the environmental features. Therefore, the use of FCNs seems to be a handy and appropriate tool for robot social navigation. A comparison of different network architectures could be considered, and a modification of the process to learn in dynamic scenarios could improve the quality of the path planning in such populated scenarios.

All in all, this thesis posed a set of different methods and techniques, always from a perspective of Learning from Demonstration, applied to the problem of human-aware navigation. We aimed at capturing the underlying behaviors of human experts demonstrating social navigation tasks and transfer these behaviors to a mobile robot. The various drawbacks found in the development of the different techniques presented have been addressed with new approaches. So, the work presents a clear evolution from the first approaches to the latest ones.

9.2 Future work

The methods and approximations developed in this thesis still present some questions and hindrances to be addressed in future work. We indicate them here along with other open issues in the general problem of human-aware navigation of mobile robots.

A common and remarkable issue of the approximations proposed in this thesis is that they learn from demonstrations in static situations where the pedestrians are still, and dynamic features of the environment are not taken into account. In this sense, the local motion controller learned in Chapter 3 could be extended with features related to the pedestrian velocity. Moreover, the addition of time and other velocity-related descriptors to the unsupervised learning proposed in Chapter 5 could be explored. In the case of the learning algorithm presented in Chapter 6, the use of features related to pedestrians is not enough. How to provide the demonstrations and how to reproduce the situations during the learning process need to be determined. One option could be to split the demonstrations in sequences that are then reproduced backward during the learning. Finally, the approach based on Fully Convolutional Networks, Chapter 8, could be adapted to dynamic scenarios by exploring the use of recurrent networks and providing the demonstrations as sequences of images regarding different time steps of the trajectory. In particular, the use of Long short-term memories (LSTM) could be studied. They are networks with loops in them, allowing information to persist. This can help to predict the pedestrians' trajectories and plan the robot's path accordingly.

In this sense, new works on the prediction of pedestrians' movement in the computer vision community using deep neural networks should be considered (Yi et al., 2016; Su et al., 2016, 2017). Other relevant references based on modeling

the pedestrian's trajectories are (Kim et al., 2015), or (Karamouzas et al., 2014) in which a model crowd prediction is derived from data.

Regarding this latter approach, the use of FCN to learn the relevant features involved in the task from nearly raw sensory input delivers promising results. The study of different networks architectures and the comparison of their results can be considered. Moreover, the recent Deep-IRL concepts, like the Maximum Entropy Deep IRL approach (Wulfmeier et al., 2015, 2016), could be employed to modify the learning algorithm presented in Chapter 6 to remove the dependency on the manually-designed RRT* cost function.

Another point that can be further studied is the performance of the RRT* path planner. Through this thesis, a 2D no-kinodynamic version of the RRT* algorithm has been extensively used. The need of fast re-planning in the changing social environments did not allow us to include the kinodynamic constraints in the path planning phase what increases the computational cost significantly. Some steps in that direction were taken in the approximations presented in Chapters 5 and 8, in which we partially reduced the sampling space of the RRT* to areas of interest, therefore removing unnecessary computations. However, this needs to be further studied and improved to be able to compute the kinodynamic constraints at high frequency. Considering the kinodynamic constraints allows to predict robot positions in concrete future steps, and therefore, it opens new possibilities to plan "social" trajectories.

Finally, although it is not directly addressed in this thesis, there are still some other general open issues in the field of human-aware navigation that are worthy to mention.

One of them is the perception of people in the robot's vicinity in real environments. The detection of a person is not enough to perform good "social" navigation. Determining the correct body orientation and also the head orientation can give important information to infer the person's intentions and act accordingly. Many works consider the use of artificial markers or the deployment in very controlled environments to provide a correct detection. For instance, the method for approaching people presented in Chapter 5, was successfully tested by using a motion capture system that provided accurate people positions and orientations to the robot. By contrast, with the TERESA detection and tracking system onboard the robot, we had problems to detect the correct orientation of the person to approach, which led to a bad approaching behavior under real uncontrolled conditions. Therefore, more accurate and advanced perception systems that can be executed on-board are required to fulfill the goal.

Another problem is the robot motion. As we experienced with the FROG robot, and partially with TERESA, in Chapters 4 and 7, social robots need to be able to quickly adapt to the dynamic conditions of the environment with rapid

and also smooth movements. One the one hand, fast computational algorithms able to process all the sensors data and plan the robot motion in a few milliseconds are required. On the other hand, agile kinematics and a lighter mechanical design are also needed. Faster and more accurate actuators will improve the robot response, and omnidirectional kinematics seems to ease the smoothness and agility of the robot.

Bibliography

P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015430.

B.D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstrations. *Robotics and Autonomous Systems*, 57:469–483, 2009.

K.O. Arras, O. Martinez-Mozos, and W. Burgard. Using boosted features for the detection of people in 2D range data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3402–3407, Roma, Italy, April 2007. ISBN 1-4244-0602-1.

O. Arslan and D. E. Koditschek. Exact robot navigation using power diagrams. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, May 2016a. doi: 10.1109/ICRA.2016.7487090.

O. Arslan and D.E. Koditschek. Sensor-based reactive navigation in unknown convex sphere worlds. In *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016b.

F. Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. doi: 10.1137/0216006.

M. Barnaud, N. Morgado, R. Palluel-germain, J. Diard, and A. Spalanzani. Proxemics models for human-aware navigation in robotics : Grounding interaction and personal space models in experimental data from psychology. *Proceedings of the 3rd IROS'2014 workshop Assistance and Service Robotics in a Human Environment*, 2014.

S. Behnke, F. Faber, M. Bennewitz, A. Görög, C. Gonsior, D. Joho, and M. Schreiber. The humanoid museum tour guide Robotinho. In *in IEEE*

Int. Symp. on Robot and Human Interactive Communication, 2009. ISBN 9781424450817.

M. Bennewitz. *Mobile Robot Navigation in Dynamic Environments.* PhD thesis, University of Freiburg, Department of Computer Science, 2004.

M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005. doi: 10.1177/0278364904048962.

P. Bhattacharya and M. L. Gavrilova. Voronoi diagram in optimal path planning. In *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pages 38–47, July 2007. doi: 10.1109/ISVD.2007.43.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Robot Programming by Demonstration*, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_60.

S. Bitgood and S. Dukes. Not another step! economy of movement and pedestrian choice point behavior in shopping malls. *Environment and Behavior*, 38(3): 394–405, 2006. doi: 10.1177/0013916505280081.

J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, Jun 1991. ISSN 1042-296X. doi: 10.1109/70.88137.

G. Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, June 1986. ISSN 0734-189X. doi: 10.1016/S0734-189X(86)80047-0.

O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *In IEEE Int. Conf. on Robotics and Automation*, pages 341–346, 1999.

R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

W. Burgard, Armin B. Cremers, D. Fox, D. Hhnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1):3 – 55, 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00070-3.

S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach.* EPFL/CRC Press, 2009. EPFL Press ISBN 978-2-940222-31-5, CRC Press ISBN 978-1-4398-0867-2.

S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008.

S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(2):286–298, 2007.

Yu Fan Chen, M. Everett, Miao Liu, and Jonathan P. How. Socially aware motion planning with deep reinforcement learning. *CoRR*, abs/1703.08862, 2017a.

Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P. How. Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement Learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2017b.

J. Claassens. A RRT-based path planner for use in trajectory imitation. In *Proc. of the International Conference on Robotics and Automation, ICRA*, pages 3090–3095. IEEE, 2010.

J. Cortés, L. Jaillet, and T. Siméon. Molecular disassembly with rrt-like algorithms. In *ICRA*, pages 3301–3306. IEEE, 2007.

M. Cristani, L. Bazzani, G. Paggetti, A. Fossati, D. Tosato, A. Bue, G. Menegaz, and V. Murino. Social interaction discovery by statistical analysis of F-formations. *British Machine Vision Conference (BMVC)*, pages 23.1–23.12, 2011. doi: 10.5244/C.25.23.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

R. Dennis Middlemist, E. Knowles, and C. F. Matter. Personal space invasions in the lavatory: Suggestive evidence for arousal. *Journal of personality and social psychology*, 33:541–6, 06 1976.

M. Desai, K.M. Tsui, H.A. Yanco, and C. Uhlik. Essential features of telepresence robots. In *Technologies for Practical Robot Applications (TePRA), 2011 IEEE Conference on*, pages 15–20, April 2011. doi: 10.1109/TEPRA.2011.5753474.

D. Devaurs, T. Simeon, and J. Cortes. Enhancing the transition-based RRT to deal with complex cost spaces. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4120–4125, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6631158.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390.

Yong Duan, Baoxia Cui, and Huaiqing Yang. *Robot Navigation Based on Fuzzy RL Algorithm*, pages 391–399. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-87732-5. doi: 10.1007/978-3-540-87732-5_44.

D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2366–2374, Cambridge, MA, USA, 2014. MIT Press.

V. Evers, N. Menezes, L. Merino, D. Gavrilla, F. Nabais, M. Pantic, P. Alvito, and D. Karreman. *The Development and Real-World Deployment of FROG, the Fun Robotic Outdoor Guide*, pages 100–100. ACM, 3 2014. ISBN 978-1-4503-2658-2. doi: 10.1145/2559636.2559649. eemcs-eprint-25504.

D. Ferguson and A. Stentz. The field d* algorithm for improved path planning and replanning in uniform and non-uniform cost environments. Technical Report CMU-RI-TR-05-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2005.

G. Ferrer and A. Sanfeliu. Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1730–1735, Sept 2014. doi: 10.1109/IROS.2014.6942788.

G. Ferrer and A. Sanfeliu. Multi-objective cost-to-go functions on robot navigation in dynamic environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3824–3829, Sept 2015. doi: 10.1109/IROS.2015.7353914.

G. Ferrer, A. Garrell, and A. Sanfeliu. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1688–1694, Nov 2013. doi: 10.1109/IROS.2013.6696576.

G. Ferrer, A. Garrel, F. Herrero, and A. Sanfeliu. Robot social-aware navigation framework to accompany people walking side-by-side. *Autonomous Robots*, pages 1–19, 2016. ISSN 1573-7527. doi: 10.1007/s10514-016-9584-y.

P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 560–565 vol.1, May 1993. doi: 10.1109/ROBOT.1993.292038.

P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, 4(1), 1997.

C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay. Risk based motion planning and navigation in uncertain dynamic environment. Research report, Inria, October 2010.

Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *IEEE International Conference on Intelligent Robots and Systems*, pages 2997–3004, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6942976.

S. Garrido, M. Abderrahim, and L. Moreno. Path planning and navigation using voronoi diagram and fast marching. *IFAC Proceedings Volumes*, 39(15):346 – 351, 2006. ISSN 1474-6670. doi: https://doi.org/10.3182/20060906-3-IT-2910.00059. 8th IFAC Symposium on Robot Control.

B.P. Gerkey and K. Konolige. Planning and control in unstructured terrain. In *In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 2008.

E. Goffman. *Relations in public; microstudies of the public order*. Basic Books New York, 1971. ISBN 0465068952.

D. Gonzalez-Bautista, J. Pérez, V. Milanes, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 11 2015.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

H. Gross, H. Boehme, C. Schrter, S. Mller, 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 *IROS*, pages 2005–2012. IEEE, 2009. ISBN 978-1-4244-3804-4.

J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro. Human-friendly robot navigation in dynamic environments. In *2013 IEEE International Conference on Robotics and Automation*, pages 423–430, May 2013. doi: 10.1109/ICRA.2013.6630610.

E.T. Hall. *The Hidden Dimension*. Anchor, October 1966. ISBN 0385084765.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995. doi: 10.1103/PhysRevE.51.4282.

P. Henry, C. Vollmer, B. Ferris, and D. Fox. Learning to navigate through crowded environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 981–986, 2010.

M. Herman, V. Fischer, T. Gindele, and W. Burgard. Inverse reinforcement learning of behavioral models for online-adapting navigation strategies. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3215–3222, 2015. ISSN 10504729. doi: 10.1109/ICRA.2015.7139642.

T. Hester, M. Quinlan, and P. Stone. Generalized model learning for reinforcement learning on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2369–2374, May 2010. ISBN 9781424450404. doi: 10.1109/ROBOT.2010.5509181.

T. Hester, M. Quinlan, and P. Stone. A Real-Time Model-Based Reinforcement Learning Architecture for Robot Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

O. Islas-Ramirez, H. Khambhaita, R. Chatila, M. Chetouani, and R. Alami. Robots learning how and where to approach people. In *RO-MAN 2016 25th, IEEE International Symposium on Robot and Human Interactive Communication*, 2016.

L. Jaillet, J. Cortés, and T. Siméon. Transition-based RRT for path planning in continuous cost spaces. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 2145–2150, 2008. doi: 10.1109/IROS.2008.4650993.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

I. Karamouzas, B. Skinner, and S.J. Guy. Universal power law governing pedestrian interactions. *Phys. Rev. Lett.*, 113:238701, Dec 2014. doi: 10.1103/PhysRevLett.113.238701.

D. Karreman. *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies*. PhD thesis, University of Twente, 9 2016.

D. Karreman, L. Utama, M.P. Joosse, M. Lohse, E. van Dijk, and V. Evers. *Robot etiquette: How to approach a pair of people?*, pages 196–197. ACM, 3 2014. ISBN 978-1-4503-2658-2. doi: 10.1145/2559636.2559839. eemcs-eprint-25423.

L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, pages 566–580, 1996.

H. Khambhaita. *Human-aware space sharing and navigation for an interactive robot*. PhD thesis, University of Toulouse, 10 2017.

H. Khambhaita and R. Alami. A human-robot cooperative navigation planner. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 161–162, New York, NY, USA, 2017a. ACM. ISBN 978-1-4503-4885-0. doi: 10.1145/3029798.3038374.

H. Khambhaita and R. Alami. Assessing the Social Criteria for Human-Robot Collaborative Navigation: A Comparison of Human-Aware Navigation Planners. In *Proc. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, page 6p., Lisbonne, Portugal, August 2017b.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, April 1986. ISSN 0278-3649. doi: 10.1177/027836498600500106.

O. Khatib, H. Jaouni, R. Chatila, and J. P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 2920–2925 vol.4, Apr 1997. doi: 10.1109/ROBOT.1997.606730.

B. Kim and J. Pineau. Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning. *International Journal of Social Robotics*, 8(1):51–66, 2016. ISSN 18754805. doi: 10.1007/s12369-015-0310-2.

S. Kim, S.J. Guy, W. Liu, D. Wilkie, R.W.H. Lau, M.C Lin, and D. Manocha. Brvo: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research*, 34(2):201–217, 2015.

R. Kirby, R.G. Simmons, and J. Forlizzi. Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN*, pages 607–612. IEEE, 2009.

R. Kirby, J. J. Forlizzi, and R. Simmons. Affective social robots. *Robotics and Autonomous Systems*, 58:322–332, 2010.

B. Kluge and E. Prassler. Reflective navigation: individual behaviors and group behaviors. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 4172–4177 Vol.4, April 2004. doi: 10.1109/ROBOT.2004.1308926.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. ISSN 0278-3649. doi: 10.1177/0278364913495721.

M. Kobilarov. Cross-entropy randomized motion planning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011. doi: 10.15607/RSS.2011.VII.022.

Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012. doi: 10.1177/0278364912444543.

D.E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11(4):412 – 442, 1990. ISSN 0196-8858. doi: https://doi.org/10.1016/0196-8858(90)90017-S.

S. Koenig and M. Likhachev. D*lite. In *Eighteenth National Conference on Artificial Intelligence*, pages 476–483, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0.

Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1398–1404 vol.2, Apr 1991. doi: 10.1109/ROBOT.1991.131810.

H. Kretzschmar, M. Kuderer, and W. Burgard. Inferring navigation policies for mobile robots from demonstrations. In *Proc. of the Autonomous Learning Workshop at the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

H. Kretzschmar, M. Kuderer, and W. Burgard. Learning to predict trajectories of cooperatively navigating agents. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4015–4020. IEEE, 2014.

H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 2016. doi: 10.1177/0278364915619772.

A. Kristoffersson, S. Coradeschi, and A. Loutfi. A review of mobile robotic telepresence. *Adv. in Hum.-Comp. Int.*, 2013:3:3–3:3, January 2013. ISSN 1687-5893.

T. Kruse, P. Basili, S. Glasauer, and A. Kirsch. Legible robot navigation in the proximity of moving humans. In *ARSO*, pages 83–88. IEEE, 2012. ISBN 978-1-4673-0481-8.

T. Kruse, A. Kumar Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726 – 1743, 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2013.05.007.

M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Proc. of Robotics: Science and Systems (RSS)*, Sydney, Australia, 2012.

M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Seattle, USA*, volume 134, 2015.

R. Kummerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 32(4):565–589, 2015. ISSN 1556-4967. doi: 10.1002/rob.21534.

V. Kunchev, L. Jain, V. Ivancevic, and A. Finn. *Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review*, pages 537–544. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-46539-3. doi: 10.1007/11893004_70.

Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. A. Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Contr. Sys. Techn.*, 17(5):1105–1118, 2009. doi: 10.1109/TCST.2008.2012116.

J.C. Latombe. *Exact Cell Decomposition*, pages 200–247. Springer US, Boston, MA, 1991a. ISBN 978-1-4615-4022-9. doi: 10.1007/978-1-4615-4022-9_5.

J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991b. ISBN 079239206X.

Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

Steven M. Lavalle, James J. Kuffner, and Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, 2012.

S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Neural Information Processing Systems Conference*, 2011.

S. G. Loizou. The navigation transformation. *IEEE Transactions on Robotics*, PP(99):1–8, 2017. ISSN 1552-3098. doi: 10.1109/TRO.2017.2725323.

T. Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979. ISSN 0001-0782. doi: 10.1145/359156.359164.

M. Luber, L. Spinello, J. Silva, and K. O. Arras. Socially-aware robot navigation: A learning approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 902–907, Oct 2012. doi: 10.1109/IROS.2012.6385716.

R. Mead and M.J. Matarić. Autonomous human-robot proxemics: socially aware navigation based on interaction potential. *Autonomous Robots*, pages 1–13, 2016. ISSN 0929-5593. doi: 10.1007/s10514-016-9572-2.

D. Mehta, G. Ferrer, and E. Olson. Autonomous navigation in dynamic social environments using multi-policy decision making. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1190–1197, Oct 2016. doi: 10.1109/IROS.2016.7759200.

B. Michini, M. Cutler, and J. P. How. Scalable reward learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. In *AAAI*, pages 1177–1183. AAAI Press, 2007. ISBN 978-1-57735-323-2.

A.Y. Ng and S.J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

B. Okal and K.O. Arras. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 2889–2895, 2016a. doi: 10.1109/ICRA.2016.7487452.

B. Okal and K.O. Arras. Formalizing normative robot behavior. In *Social Robotics: 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016 Proceedings*, pages 62–71. Springer International Publishing, 2016b. ISBN 978-3-319-47437-3. doi: 10.1007/978-3-319-47437-3{_}7.

B. Okal, H. Gilbert, and K.O. Arras. Efficient inverse reinforcement learning using adaptive state-graphs. In *Learning from Demonstration: Inverse Optimal Control, Reinforcement Learning and Lifelong Learning Workshop at Robotics: Science and Systems (RSS)*, Rome, Italy, 2015.

M. Otte and N. Correll. C-forest: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806, June 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2240176.

M. Otte and E. Frazzoli. RRTX: Real-time motion planning/replanning for environments with unpredictable obstacles. *Springer Tracts in Advanced Robotics*, 107:461–478, 2015. ISSN 1610742X. doi: 10.1007/978-3-319-16595-0_27.

E. Pacchierotti, H.I. Christensen, and P. Jensfelt. Evaluation of passing distance for social robots. In *IEEE Workshop on Robot and Human Interactive Communication (ROMAN)*, Hartfordshire, UK, September 2006.

L. Palmieri, S. Koenig, and K.O. Arras. RRT-based nonholonomic motion planning using any-angle path biasing. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:2775–2781, 2016. ISSN 10504729. doi: 10.1109/ICRA.2016.7487439.

S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *International Conference on Computer Vision*, 2009.

J. Pérez, F. Caballero, and L. Merino. Enhanced monte carlo localization with visual place recognition for robust robot localization. *Journal of Intelligent & Robotic Systems*, 80(3):641–656, Dec 2015. ISSN 1573-0409. doi: 10.1007/s10846-015-0198-y.

N. Pérez-Higueras, R. Ramon-Vigo, F. Caballero, and L. Merino. Robot local navigation with learned social cost functions. In *Proc. of the 11th International Conference on Informatics in Control, Automation and Robotics, ICINCO*, volume 02, pages 618–625, 2014.

N. Pérez-Higueras, F. Caballero, and L. Merino. Learning robot navigation behaviors by demonstration using a rrt* planner. In *International Conference on Social Robotics*, pages 1–10. Springer International Publishing, 2016a.

N. Pérez-Higueras, R. Ramón-Vigo, I. Perez-Hurtado, J. Capitán, F. Caballero, and L. Merino. A social navigation system in telepresence robots for elderly. In *Workshop Using Social Robots to Improve the Quality of Life in the Elderly. International Conference on Social Robotics*, 2016b.

N. Pérez-Higueras, F. Caballero, and L. Merino. Teaching robot navigation behaviors to optimal rrt planners. *International Journal of Social Robotics*, Nov 2017. ISSN 1875-4805. doi: 10.1007/s12369-017-0448-1.

N. Pérez-Higueras, F. Caballero, and L. Merino. Learning human-aware path planning with fully convolutional networks. In *the IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018. To appear.

M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *In Proceedings of the International Conference on Robotics and Automation*, pages 802–807, 1993.

D. Ramachandran and E. Amir. Bayesian Inverse Reinforcement Learning. *Proceedings of the 20th International Joint Conference on Artical Intelligence*, 51: 2586–2591, 2007.

R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. Transferring human navigation behaviors into a robot local planner. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN*, 2014. doi: 10.1109/ROMAN.2014.6926347.

R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. Analyzing the relevance of features for a social navigation task. In L.P. Reis, A.P. Moreira, P. Lima, L. Montano, and V. Munoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, volume 418 of *Advances in Intelligent Systems*

and Computing, pages 235–246. Springer International Publishing, 2015. ISBN 978-3-319-27148-4. doi: 10.1007/978-3-319-27149-1_19.

R. Ramón-Vigo, N. Pérez-Higueras, F. Caballero, and L. Merino. A framework for modelling local human-robot interactions based on unsupervised learning. In *International Conference on Social Robotics*, pages 32–41. Springer International Publishing, 2016.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006a.

C.E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006b.

N.D. Ratliff, J.A. Bagnell, and M. Zinkevich. Maximum margin planning. *International conference on Machine learning - ICML '06*, 23:729–736, 2006. ISSN 17458358. doi: 10.1145/1143844.1143936.

N.D. Ratliff, D. Silver, and J.A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009. ISSN 1573-7527. doi: 10.1007/s10514-009-9121-3.

J. Rios-Martinez, A. Spalanzani, and C. Laugier. Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2014–2019, Sept 2011. doi: 10.1109/IROS.2011.6094496.

C. Rosmann, W. Feiten, T. Wsch, F. Hoffmann, and T. Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143, Sept 2013. doi: 10.1109/ECMR.2013.6698833.

C. Rosmann, F. Hoffmann, and T. Bertram. Planning of multiple robot trajectories in distinctive topologies. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, Sept 2015. doi: 10.1109/ECMR.2015.7324179.

S. Satake, T. Kanda, D. F. Glas, M. Imai, H. Ishiguro, and N. Hagita. A robot that approaches pedestrians. *IEEE Transactions on Robotics*, 29(2):508–524, April 2013. ISSN 1552-3098. doi: 10.1109/TRO.2012.2226387.

S. Satake, K. Hayashi, K. Nakatani, and T. Kanda. Field trial of an information-providing robot in a shopping mall. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1832–1839, Sept 2015. doi: 10.1109/IROS.2015.7353616.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6: 461–464, 1978.

F. Setti, C. Russell, C. Bassetti, and M. Cristani. F-formation detection: Individuating free-standing conversational groups in images. *PLoS ONE*, 10(5): 1–32, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0123783.

S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers. Learning to Drive using Inverse Reinforcement Learning and Deep Q-Networks. In *NIPS workshop on Deep Learning for Action and Interaction*, 2016.

K. Shiarlis, J. Messias, M. van Someren, S. Whiteson, J Kim, J Vroon, G. Englebienne, K. Truong, V. Evers, N. Pérez-Higueras, I. Perez-Hurtado, R. Ramón-Vigo, F. Caballero, L. Merino, J. Shen, S. Petridis, M. Pantic, L. Hedman, M. Scherlund, R. Koster, and H. Michel. Teresa: A socially intelligent semi-autonomous telepresence system. In *Workshop on Machine Learning for Social Robotics at the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

K. Shiarlis, J. Messias, and S. Whiteson. Rapidly exploring learning trees. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, May 2017a. IEEE.

K. Shiarlis, J. Messias, and S. Whiteson. Acquiring social interaction behaviours for telepresence robots via deep learning from demonstration. In *IROS 2017: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2017b.

Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-23957-4. doi: 10.1007/978-3-540-30301-5.

E.A. Sisbot, L.F. Marin-Urias, R. Alami, and T. Siméon. A Human Aware Mobile Robot Motion Planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007. doi: 10.1109/TRO.2007.904911.

William D. Smart. *Making Reinforcement Learning Work on Real Robots*. PhD thesis, Department of Computer Science at Brown University, Providence, RI, USA, 2002. AAI3050968.

A. Spalanzani, J. Rios-Martinez, C. Laugier, and S. Lee. *Risk Based Navigation Decisions*, pages 1459–1477. Springer London, London, 2012. ISBN 978-0-85729-085-4. doi: 10.1007/978-0-85729-085-4_56.

P. Stein, V. Santos, A. Spalanzani, and C. Laugier. Navigating in populated environments by following a leader. In *2013 IEEE RO-MAN*, pages 527–532, Aug 2013. doi: 10.1109/ROMAN.2013.6628558.

P. Stein, V. Santos, A. Spalanzani, and C. Laugier. Learning-Based Leader Classification. In *IROS 2013 - Workshop on Assistance and Service Robotics*, Tokyo, Japan, November 2014.

A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 3310–3317 vol.4, May 1994. doi: 10.1109/ROBOT.1994.351061.

Hang Su, Yinpeng Dong, Jun Zhu, Haibin Ling, and Bo Zhang. Crowd scene understanding with coherent recurrent neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3469–3476. AAAI Press, 2016. ISBN 978-1-57735-770-4.

Hang Su, Jun Zhu, Yinpeng Dong, and Bo Zhang. Forecast the plausible paths in crowd scenes. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI-17)*, pages 2772–2778, 2017. doi: 10.24963/ijcai.2017/386.

C. Tay and C. Laugier. Modelling Smooth Paths Using Gaussian Processes. In *Proc. of the Int. Conf. on Field and Service Robotics*, Chamonix, France, 2007.

TERESA Consortium. Deliverable D6.1: Requirements Report, 2014. `http://teresaproject.eu/wp-content/uploads/2015/03/D6_1_Requirements-Report_-TERESA1.pdf`.

TERESA Consortium. Deliverable D2.2: Pose Estimation, 2016a. `http://teresaproject.eu/wp-content/uploads/2016/06/D2.2_Localisation.pdf`.

TERESA Consortium. Deliverable D6.6: Evaluation report, 2016b.

S. Thrun. An approach to learning mobile robot navigation. *Robotics and Autonomous Systems*, 15(4):301 – 319, 1995. ISSN 0921-8890. doi: 10.1016/0921-8890(95)00022-8. Reinforcement Learning and Robotics.

S. Thrun, M. Bennewitz, W. Burgard, Armin B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A second generation mobile tour-guide robot. In *International Conference on Robotics and Automation (ICRA)*, 1999.

P. Trautman. Assistive planning in complex, dynamic environments: A probabilistic approach. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3072–3078, Oct 2015. doi: 10.1109/SMC.2015.534.

P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of humanrobot cooperation. *The International Journal of Robotics Research*, 34(3), 2015. ISSN 10504729. doi: 10.1177/0278364914557874.

R. Triebel, K.O. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang. SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports. *Field and Service Robotics*, pages 1–14, 2015. ISSN 1610742X. doi: 10.1007/978-3-319-27702-8_40.

Xuan-Tung Truong and Trung-Dung Ngo. Dynamic social zone based mobile robot navigation for human comfortable safety in social environments. *International Journal of Social Robotics*, pages 1–22, 2016. ISSN 1875-4805. doi: 10.1007/s12369-016-0352-0.

K.M. Tsui, M. Desai, H.A. Yanco, and C. Uhlik. Exploring use cases for telepresence robots. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pages 11–18, March 2011.

A. Turnwald, D. Althoff, D. Wollherr, and M. Buss. Understanding Human Avoidance Behavior: Interaction-Aware Decision Making Based on Game Theory. *International Journal of Social Robotics*, 8(2):331–351, 2016. ISSN 18754805. doi: 10.1007/s12369-016-0342-2.

J. van den Berg, S.J. Guy, M.C. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH*, 2009.

D. Vasquez, T. Fraichard, and C. Laugier. Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. *The International Journal of Robotics Research*, 28(11-12):1486–1506, 2009. doi: 10.1177/0278364909342118.

D. Vasquez, B. Okal, and K.O. Arras. Inverse Reinforcement Learning Algorithms and Features for Robot Navigation in Crowds: an experimental comparison. *Proc. IEEE/RSJ Int. Conference on Intelligent Robots*

and Systems (IROS), 2014, pages 1341–1346, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6942731.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, Feb 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1167.

Z. Wang, K. Mlling, M.P. Deisenroth, H. Ben Amor, D. Vogt, B. Schlkopf, and J. Peters. Probabilistic movement modeling for intention inference in humanrobot interaction. *The International Journal of Robotics Research*, 32 (7):841–858, 2013. doi: 10.1177/0278364913478447.

M. Wulfmeier, P. Ondruska, and I. Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

M. Wulfmeier, Dominic Z. Wang, and I. Posner. Watch This: Scalable Cost-Function Learning for Path Planning in Urban Environments . In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

Chen Xia and Abdelkader El Kamel. Neural inverse reinforcement learning in autonomous navigation. *Robotics and Autonomous Systems*, 84:1–14, 2016. doi: 10.1016/j.robot.2016.06.003.

Shuai Yi, Hongsheng Li, and Xiaogang Wang. *Pedestrian Behavior Understanding and Prediction with Deep Neural Networks*, pages 263–279. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46448-0. doi: 10.1007/978-3-319-46448-0_16.

F. Zanlungo, T. Ikeda, and T. Kanda. A Microscopic "Social Norm" Model to Obtain Realistic Macroscopic Velocity and Density Pedestrian Distributions. *PLoS ONE*, 7(12), 2012. ISSN 19326203. doi: 10.1371/journal.pone.0050720.

B. Ziebart, A. Maas, J. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2008.

M. Zucker, J. J. Kuffner, and M. S. Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pages 1603–1609, 2007. doi: 10.1109/ROBOT.2007.363553.